University of Alberta

Active Reliable Multicast with Congestion Control

by

Baochun Bai

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

Department of Computing Science

Edmonton, Alberta
Spring 2002

National Library of Canada
Acquisitions and Bibliographic Services
395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada
Acquisitions et services bibliographiques
395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-69682-0

Canada

University of Alberta

Library Release Form

Name of Author: Baochun Bai

Title of Thesis: Active Reliable Multicast with Congestion Control

Degree: Master of Science

Year this Degree Granted: 2002

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as herein before provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

Baochun Bai
Department of Computing Science
232 Athabasca Hall
University of Alberta
Edmonton, AB
Canada, T6G 2E8

Date: Jan. 2, 2002

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Active Reliable Multicast with Congestion Control** submitted by Baochun Bai in partial fulfillment of the requirements for the degree of **Master of Science**.

Dr. Janelle Harms

Dr. Ioanis Nikolaidis

Dr. Marek Reformat

Date: Dec 17, 2001

Abstract

In this thesis, we propose a novel active reliable multicast protocol with congestion control (ARMC). Our protocol is a complete one that includes both an error recovery scheme and a congestion control scheme. The ARMC uses soft-state storage to facilitate fast error recovery and efficient congestion control. The error recovery scheme includes five mechanisms: a simple NAK suppression and aggregation mechanism, an efficient hierarchical RTT measurement mechanism, an effective local recovery and scoped retransmission mechanism, and a periodical ACK mechanism. In a radical departure from popular end-to-end congestion control, our congestion control scheme is a router-assisted window-based hierarchical scheme. With flexible use of the resources in active routers, our congestion control scheme can not only let multicast flows have an adjustable share of bandwidth of bottleneck links with TCP flows, but also let two different multicast flows have a different share of bandwidth of bottleneck links according to the resources they (are allowed to) use. In addition, our congestion control scheme can also achieve inter-receiver fairness [TM98], [TE99] by limiting the effect of congestion on a specific link to a small region. ARMC can also achieve scalability and responsiveness. We implement and evaluate our protocol in NS2 and also compare performance with ARM [LS98] and AER/NCA [SS00], both of which are representative active reliable multicast protocols. The initial performance evaluation indicates that our protocol can achieve considerable performance improvement with limited support from routers. Our work further confirms that active networks can benefit some applications and become a promising network computing platform in the future.

Acknowledgements

I would like to express my gratitude to my supervisor Dr. Janelle Harms, for her thoughtful discussion on technical problems and generous financial support. I have been very fortunate to have both her guidance and encouragement during my graduate study. I would also like to thank the members of my examining committee: Dr. Ioanis Nikolaidis and Dr. Marek Reformat, for their review in this work.

I am grateful to Dr. Mike MacGregor and Dr. Paul Lu. Their kind help makes me stay here so pleasurable. I am also indebted to my good friend and teammate, Yuxi Li, for the great cooperation with him.

Finally I would like to thank my fellow graduate students in network research lab, Yanxia Jia, Chong Zhong, Xiaoqin Yuan, Weiguang Shi, Justin Gamble, Kui Wu, Xudong Wu, Hongjun Zhang, Ioannis Batsiolas, Juhua Shi, Qiang Ye, and Xiang Wan for their friendship.

Contents

1 Introduction **1**

1.1 Multicast 1

1.2 Active Networks and End-to-end Argument 3

1.3 Thesis Contribution 4

1.4 Thesis Outline 5

2 Related Work **6**

2.1 Error Recovery 6

2.1.1 Challenges for error recovery 6

2.1.2 Randomized backoff schemes 7

2.1.3 Hierarchical schemes 8

2.1.4 Other schemes 13

2.2 Congestion Control 14

2.2.1 Challenges for congestion control 14

2.2.2 Fairness issues 15

2.2.3 Single-rate schemes 17

2.2.4 Multi-rate schemes 20

2.3 Summary 22

3 Detailed Description of ARMC **23**

3.1 Network Environment 23

3.2 Protocol Overview 24

3.3 The Error Recovery Scheme of ARMC 25

3.3.1 Overview 26

3.3.2 Early packet loss detection 28

3.3.3 NAK suppression and aggregation 28

3.3.4 Local recovery and scoped retransmission 31

3.3.5 Hierarchical RTT measurement 31
3.3.6 Periodical ACK mechanism 34
3.3.7 Buffer management strategy 34
3.3.8 Scalability issues 34
3.4 Congestion Control 35
3.4.1 Overview 35
3.4.2 Design objectives 36
3.4.3 Congestion detection 39
3.4.4 Start and termination of congestion smoothing phase . 41
3.4.5 Congestion window regulation algorithm 41
3.4.6 Congestion smoothing buffer management 43
3.4.7 Handling drop-to-zero problem 47
3.4.8 Handling heterogeneous link bandwidths and receivers 48
3.4.9 Possible limitations 49
3.5 Relationship between Error Recovery and Congestion Control 50
3.6 Handling Dynamic Membership Changes 50
3.7 Summary 51

4 Performance Evaluation 52
4.1 Validation and Verification of Simulation 52
4.2 Sensitivity Test 52
4.3 Error Recovery 55
4.3.1 Simulation configuration 55
4.3.2 Effect of retransmission buffer size on performance . . 56
4.3.3 Effect of number of active routers on performance . . . 58
4.3.4 Comparison of ARMC , ARM, and AER/NCA 67
4.3.5 Summary for error recovery 75
4.4 Congestion Control 75
4.4.1 Interaction with TCP flows 75
4.4.2 Interaction among multiple multicast flows 86
4.4.3 Inter-receiver fairness 96
4.4.4 Summary for congestion control 101
4.5 Summary 101

5 Conclusions and Future Work **102**

5.1 Conclusions 102

5.2 Future Work 103

Bibliography **105**

List of Figures

3.1 Error Recovery 29
3.2 Hierarchical RTT measurement 32
3.3 Congestion control and avoidance 37
3.4 Rate Regulation Algorithm at an Active Router 42
3.5 Rate Regulation Algorithm at the Source 42
3.6 Analysis Topology 44
3.7 Rate and Buffer Change during the Congestion Smooth Duration 45
3.8 drop-to-zero problem 47

4.1 RTT curve 54
4.2 Average recovery time in 35% active routers case 60
4.3 Nuisance in 35% active routers case 62
4.4 Average recovery time in buffer size = 40 64
4.5 Nuisance in buffer size = 40 66
4.6 Average recovery time for comparison of ARMC, ARM and AER/NCA in 35% active routers and buffer size = 40 70
4.7 Nuisance for comparison of ARMC, ARM and AER/NCA in 35% active routers and buffer size = 40 72
4.8 Topology 76
4.9 Receiving rate when a multicast flow and a TCP flow interact along the longest path. A multicast source has the same RTT as the TCP flow. 78
4.10 Average receiving rate when a multicast flow and a TCP flow interact along the longest path. A multicast source has the same RTT as the TCP flow. 80
4.11 Receiving rate when a multicast flow and a TCP flow interact along the non-longest path. A multicast source has different RTT from the TCP flow. 83

4.12 Average receiving rate when a multicast flow and a TCP flow interact along the non-longest path. A multicast source has different RTT from the TCP flow. 85
4.13 Topology 86
4.14 Receiving rate when two multicast flows use the same amount of resources. 89
4.15 Average receiving rate when two multicast flows use the same amount of resources. 91
4.16 Receiving rate when two multicast flows use different amount of resources. 93
4.17 Average receiving rate when two multicast flows use different amount of resources. 95
4.18 Topology 96
4.19 Inter-receiver fairness. $\alpha = 2$ 98
4.20 Inter-receiver fairness. $\alpha = 4$ 100

List of Tables

4.1 Total Number of RTT_QUERY and RTT_REPLY Packets of Source 53
4.2 Number of NAKs Sent by All the Receivers 73
4.3 Number of Overhead Packets for NAK suppression 73

Chapter 1

Introduction

With the continual evolution of the Internet, group-based applications such as distributed database update and software distribution will significantly contribute to the growth of the Internet in the future. These group-based applications demand efficient reliable delivery of data packets from one source to multiple receivers. The requirements can best be satisfied by using the reliable multicast service, which has traditionally been addressed using the pure end-to-end approach. In this thesis, a novel reliable multicast service, which makes full use of the power of active networks, is presented. Active networks put functionality in the middle of the network. In this chapter, we introduce the basic concepts of multicast and active networks. We also outline our work and contribution.

1.1 Multicast

As a unicast reliable transport protocol, TCP (Transmission Control Protocol) has achieved great success and made significant contributions to the exponential growth of the Internet. Along with unicast-based applications, group-based applications such as distributed database update, multiplayer games, software distribution, and video distribution have emerged and are expected to grow significantly in the near future. The approach that uses TCP to support group-based applications has significant disadvantages: (a) duplicate packet transmission on links shared by group members wastes important net-

work resources and makes it impossible to scale to large applications which may involve millions of users, and (b) application-level synchronization among a sender and its receivers is hard to achieve when group size becomes large. Therefore, multicast protocols, whose target applications are group-based ones, are an indispensable infrastructure for the future Internet.

IP Multicast[SD90] is designed to support applications associated with a group, which is identified by a certain address (the IP class D address). A multicast group usually includes a certain number of sources and receivers. Multicast routing is responsible for establishing a tree to connect the sources and receivers and delivering the packets from the sources to receivers. As opposed to unicast delivery, multicast data packets are copied at routers inside the network rather than at the sources. Therefore, it guarantees that only one copy of a multicast data packet is transmitted per link, instead of sending one copy of the packet per receiver accessed via that link. Multicast inherently can reduce the bandwidth consumption and improve the efficiency of group applications. IP Multicast is an open service model, where no mechanism for group management and access control. Multicast datagrams are best effort and unreliable.

Reliable multicast is a basic reliable transportation service to support group-based applications. In the last decade, a number of reliable multicast protocols have been proposed [SS00, SV95, JS97, HS95, LS98, JJ00, YY00, CG98, KS99, RJ95, AS99]. Basically, a practical reliable multicast protocol should include two mechanisms: error recovery and congestion control. The error recovery scheme is an essential part of a reliable multicast protocol. The function of the error recovery mechanism is to retransmit lost packets and guarantee the reliable delivery of packets to all the receivers. To achieve low-latency packet loss detection and recovery, an error recovery mechanism has two main challenges:(a) how to avoid the NAK or ACK implosion problem at the source when the multicast group size is large, and (b) how to limit the scope of repair packets so as to keep the number of duplicate repair packets delivered to receivers as small as possible.

A congestion control and avoidance scheme is an indispensable part of a multicast protocol to support large-scale group-based applications. Multicast congestion control is a hard problem. There are two main challenges. Given the ubiquitous presence of TCP traffic in the Internet, the first challenge is how to compete for the bandwidth with TCP flows. This challenge in essence is how to define and implement "fairness" between multicast flows and unicast flows. Previous research [DJ99], [FA98], [HM98], [AJ01] discusses the fairness from different points of view. Though each of them gives an insight to the essence of a congestion control algorithm, a formal congestion control algorithm to meet the exact fairness criterion is hard to realize. Most of the proposed congestion control schemes [SS00], [LN99], [SM00], [LL98], [MS99], [LR00], [YC01] try to conform to the practical criterion of "TCP-friendliness[SK99]", which stipulates that the arrival rate of a non-TCP flow should not exceed that of a conformant TCP flow in the same circumstances. Given that a multicast session may involve a large amount of heterogeneous networks and receivers, the second challenge is how to satisfy the specific interest of heterogeneous receivers in the same multicast session that may involve heterogeneous links. This challenge is essentially how to define the "fairness" among receivers. [TM98] and [TE99] gives a definition of inter-receiver fairness, which stipulates that the receiving rate for each individual receiver should not be affected by other receivers in the same multicast group. The generic approach to address inter-receiver fairness is partitioning receivers based on their available bandwidth into multiple groups each of which has a different rate [LL98], [SV96], [SJ98], [XS98]. With a pure end-to-end approach, it is hard to address both "TCP-friendliness" and "inter-receiver fairness" for a multicast communication.

1.2 Active Networks and End-to-end Argument

Active networks [AH99] is a network architecture which can potentially become the network computing platform in the future. The original vision for active

networks is to enable new applications to leverage the computation within the network and to accelerate the introduction and deployment of new Internet services. The philosophy behind active networks is to provide more flexibility to applications without significantly compromising the traditional performance criteria such as transmission rate and security. Namely, network services can be tailored to an application's requirements by injecting specific functions to active routers for performance improvement[SK97]. At first sight, it seems to violate and conflict with the classical "end-to-end arguments [HD84]" principle for distributed computing system design. This principle stipulates that high-level functions should be implemented at the end nodes and should not be realized within a network. However, as stated in [HD84], the "end-to-end argument" is not an absolute rule, but a guideline to help decide whether it is worth implementing some high-level functions in low-level systems. It is essentially a tradeoff between performance enhancement and cost of implementation when supporting an end-to-end service in a communication system. [SK97] concludes that it is reconcilable to the end-to-end argument to provide generic services to users in an active network. Some applications, such as reliable multicast and application driven specific congestion control, can benefit from implementing some functions in networks. Previous work on active network [SS00, LS98, TF98] also confirm this conclusion. Furthermore, with active networks, [NS01] states that a new business model where network providers compete to sell their resources and users buy resources for performance enhancement can be created.

1.3 Thesis Contribution

In this thesis, we present a novel Active Reliable Multicast protocol with Congestion Control (ARMC). ARMC is a one-to-many reliable multicast protocol. ARMC includes two parts: an error recovery scheme and a congestion control scheme. It is designed to achieve good performance in an active network context. It can also be deployed in the current Internet with limited support from

routers. Most existing reliable multicast protocols only focus partially on the error recovery scheme or the congestion control scheme. Few of them combine these two parts into a complete protocol. The contributions of this thesis are listed as follows.

- The ARMC incorporates both the error recovery scheme and the congestion control scheme into a complete reliable multicast protocol.
- The ARMC innovates and integrates several simple but effective mechanisms into an improved active error recovery scheme, which makes the error recovery faster and effectively limits the retransmission scope.
- The ARMC employs a completely novel router-assisted hierarchical congestion control scheme, which can make multicast flows have an adjustable share of bandwidth of the congested links and achieve inter-receiver fairness to some degree. Meanwhile, the scalability and responsiveness can also be attained.

1.4 Thesis Outline

The rest of the thesis is organized as follows. In Chapter 2, we summarize existing related work. In Chapter 3, we present the protocol details of ARMC. In Chapter 4, we discuss our simulation results. The thesis concludes in Chapter 5. Since the proposed reliable multicast protocol includes two parts, error recovery and congestion control, we also divide each chapter into two parts and elaborate on the details.

Chapter 2

Related Work

In this chapter, we discuss the challenges of providing error recovery and congestion control for reliable multicast and some related work. We discuss them in two separate sections, error recovery and congestion control.

2.1 Error Recovery

2.1.1 Challenges for error recovery

The error recovery is an essential part for reliable multicast. Its function is to ensure that data packets are reliably delivered to every receiver. In general, the source multicasts a packet to the receivers. If a packet is lost, it must be retransmitted to the receivers that are missing it. This is called a repair packet. Feedback packets of positive acknowledgement (ACK) and negative acknowledgement (NAK) are used to facilitate this recovery. It usually has several challenges to tackle.

- **ACK implosion**: An ACK packet is the positive acknowledgment packet sent by the receivers to the source when a data packet has been successfully received. If an ACK for a receiver is not received by the source, it knows to retransmit the packet to that receiver. However, in the sender-initiated error recovery scheme that employs ACK as the feedback packet, the sender will receive ACKs from every receiver and use up its computing power and resources to handle the ACK packets particularly when the number of receivers scales to thousands even millions

of users.

- **NAK implosion**: A NAK packet is the negative acknowledgement packet sent by the receivers to the source when a data packet has been detected to be lost. In the receiver-initiated error recovery scheme that employs NAK as the feedback packet, the sender has the same problem as the sender-initiated scheme when a multicast group involves a large number of receivers.

- **Duplicate repair packets**: When some receivers experience the packet loss, if the source sends the repair packet to the whole multicast group, receivers that did not lose the packet will receive the duplicate packets. These duplicate repair packets will consume unnecessary network bandwidth and computing power of some receivers. Usually a retransmission scope mechanism need to be deployed to limit the bad effect of duplicate repair packets.

According to the mechanism to handle the ACK or NAK implosion problem, the previous error recovery schemes can basically be classified into two schemes: randomized backoff schemes and hierarchical schemes.

2.1.2 Randomized backoff schemes

Scalable Reliable Multicast (SRM) [SV95] is a representative of randomized backoff schemes. It uses a randomized backoff mechanism to avoid the NAK implosion problem. In SRM, any receiver may act as a requester to multicast a NAK to the whole group in order to find a repair packet. Also then any receiver may act as a replier to respond to the NAK with a multicast repair packet. Before transmitting a NAK or a repair packet, the requester or replier needs to wait a randomized period of time during which the transmission is suppressed and the timer is reset if an identical NAK or repair packet arrives. Session messages are used to gather host-to-host distance information that is incorporated into a randomized backoff timer to facilitate the repair mechanism and the selection of requester and replier. Any source or receiver which

owns the repair packets can respond to a NAK. The randomized backoff nature makes the recovery latency usually higher than reliable unicast mechanisms. It does not have a satisfactory scheme for retransmission scoping either.

2.1.3 Hierarchical schemes

Hierarchical schemes are a popular approach to address the ACK/NAK implosion problem and decrease duplicate repair packets. The generic solution of hierarchical scheme is to organize, either statically or dynamically, receivers into a recovery tree. NAK or ACK implosion can be controlled by aggregating or suppressing the same feedback packets in parent nodes. Scoped transmission is easily implemented by unicasting or subcasting repair packets to a specific child or to all the downstream children. Thus duplicate repair packets can be significantly decreased.

Active Reliable Multicast (ARM) [LS98] is the first reliable multicast protocol designed for active networks. In ARM, NAK is the only feedback packet and only receivers are responsible for detecting packet sequence number gaps and sending NAKs to upstream nodes. Upon receiving a NAK, active routers forward the NAK to upstream nodes and create a NAK record to enable suppression of duplicate NAKs. This is done only if it is the first NAK with a distinct packet sequence number and there is no repair packet available in this router. Otherwise, the NAK is discarded. Active routers also store the incoming data packets and repair packets in a retransmission buffer to facilitate fast error recovery. Each time an active router sends repair packets to downstream nodes, it creates a REPAIR record for suppressing the NAKs in transit that are sent by receivers before they actually receive the repair packets. Upon receiving a repair packet, active routers store it in a retransmission buffer and forward it to downstream links according to the subscription information in the NAK record and remove the NAK record if possible. If there are no available NAK records, the active router merely caches the repair packets. If the cache is used up, the repair packet is subcast to all the downstream links.

Active Error Recovery and Nominee Congestion Avoidance (AER/NCA)[SS00]

is a reliable multicast protocol using active services. AER/NCA also uses NAKs only as feedback packets. Both active routers (they call them repair servers) and receivers can detect packet sequence number gaps and send NAKs to upstream nodes. Active routers also use the retransmission buffer to store data packets and repair packets to facilitate fast recovery. However, they do not use a timer to manage the soft-state buffer. They only cache a recent set of packets in the buffer. When new data packets arrive and no free buffer is available, new packets replace the oldest packets in the buffer. AER/NCA uses a random timer mechanism to suppress duplicate NAKs. Each time when an active router or receiver wants to send a NAK, it sets a suppression timer. If it receives a NAK with the same packet sequence number before its own suppression timer goes off, it suppresses its own NAK. Otherwise, when the timer goes off, it transmits the NAK upstream unless it has received a repair packet. When an active router receives a NAK from downstream nodes, it also subcasts the NAK downstream to the subtree for NAK suppression if no repair packet is available. Otherwise, it subcasts a repair packet to all the downstream nodes. After sending or suppressing a NAK, the active router or receiver sets a retransmission timer. When the retransmission timer goes off, it can resend a NAK upstream if the NAK is not suppressed. AER/NCA uses a hierarchical RTT (Round Trip Time) measurement mechanism initiated by downstream receivers or active routers to let active routers or receivers learn the RTT value to help set the suppression timer and retransmission timer value. In AER/NCA, active routers also aggregate NAKs with the same packet sequence number. Upon receiving a repair packet, active routers subcast it to all the downstream links if there is a pending NAK record in the active router. Otherwise, they simply store it without forwarding it further down the tree. AER/NCA also includes a nominee-based end-to-end congestion control protocol, which is introduced in the next section.

The Turning Point (TP) [CG98] approach tries to achieve the optimal recovery scenario where the receiver directly below the loss sends a request to the receiver immediately above the loss which only multicasts the repair

packet to the affected branch. TP lets every on-tree router choose one of the downstream links as a *replier link* unless the router only has one downstream link. In that case, that router selects the upstream link as the replier link. Each replier link leads to a replier which is a receiver with the least loss. When a receiver detects a packet loss, it sends a request to its upstream router, which acts as a turning point and directs the request to the replier. If the request comes from the replier link, the router directly forwards the request to its upstream router. Otherwise, when a replier receives a request and it has the repair packet, it unicasts a repair packet to the router that diverted the request. When the router receives the repair packet, it subcasts the packet to all its downstream receivers.

Reliable Multicast for Core-based Multicast Routing (RCBT) [YY00] is a reliable multicast scheme designed specially for multicast routing, which uses a core-based tree to route packets to the receivers. It uses an approach similar to TP[CG98] with several improvements to cope with the error recovery issue in the core-based routing case. In a core-based tree, the upstream/downstream interface of a router is not defined with respect to the source. Therefore, a new IP option, *path_info*, is defined to carry the path information from a source to a receiver. Each receiver needs to store the path information of a specific source. When a receiver sends a request packet for error recovery, the packet should carry the stored path information to a specific source. According to the path information within the request packet, the intermediate routers can figure out the correct path leading to the source. Each router selects the replier link with respect to a specific interface. Namely, each interface in a router has a replier interface so that the router can divert the request packet from that interface to its appropriate replier link. A simple probabilistic approach is proposed to dynamically identify the most reliable replier interface according to the inter-arrival time and frequency of request packets received on an interface. To facilitate the incremental deployment of reliable multicast in the Internet, a mechanism to deal with the RCBT-incapable routers is also introduced.

Log-Based Receiver-reliable Multicast (LBRM) [HS95] achieves scalable re-

liable multicast and low-latency data recovery by organizing logging servers into a hierarchy. In particular, a two-level hierarchy is specially discussed. A primary logging server is responsible for storing (logging) all the transmitted packets from the source as long as an application desires. It sends positive feedbacks to the source to guarantee the reliability. Secondary logging servers are distributed among different sites, which are a topologically localized part of the network. A receiver requests repair packets from its local secondary logging servers rather than directly from the primary logging server. A variable heartbeat mechanism, which uses a variable interval for the source to send heartbeat packets to receivers so that they can learn the active status of the source when there are no data packets transmitted, is proposed for a receiver to easily detect the lost packets and, at the same time, reduce the overhead. A statistical acknowledgement mechanism allows the source to make intelligent choices between two error recovery approaches: re-multicast the lost data immediately or unicast retransmission by the primary logging server. The statistical technique can minimize recovery delay without imposing significant bandwidth overhead.

Tree-based Multicast Transport Protocol (TMTP) [RJ95] is a tree-based reliable multicast protocol. It organizes the group members into a hierarchy of domains. Each domain has a *domain manager* which represents the members in that domain and is responsible for error recovery within the domain. TMTP dynamically organizes the domain managers into a control tree by using an expanding ring search approach, which searches á specific node by increasing the time-to-live value (TTL) gradually to limit the propagation scope of the search packet. The hierarchical control tree is used for flow and error control. It includes both a periodic ACK feedback mechanism and a restricted NAK mechanism with NAK suppression for fast error recovery.

A Reliable Multicast Transport Protocol (RMTP) [JS97] is a one-to-many reliable multicast protocol to provide sequenced and lossless data delivery. RMTP also groups receivers into different local regions and domains. In each domain, a special receiver is chosen as a Designated Receiver(DR) to represent

this domain to send ACKs to the source and facilitate fast error recovery by local retransmission. All the DRs in a specific group are organized into a multi-level hierarchy to address the ACK implosion problem and achieve scalability. Each receiver periodically sends ACKs to the DR in its domain. If the number of retransmission requests is greater than a threshold, the DR multicasts the repair packet to its domain for error recovery. Otherwise, it unicast the repair packet to specific receivers that lost the packet. The DR also sends ACKs to its nearest upstream DR or the source so that they can multicast or unicast the repair packets according to the number of ACKs received.

Search Party[AS99] is a tree-based reliable multicast protocol proposed to improve the robustness of local recovery. It is built on a new forwarding service called *randomcast* which forwards packets randomly inside a multicast tree. When a node receives a *randomcast* packet, it randomly decides to forward the packet to its parent or another child. The probability of forwarding to its parent is computed based on a weighted distribution. The probability of forwarding the packet to a child is uniformly distributed. Search Party is enhanced from the TP[CG98] methodology. In this protocol, when a member m detects a packet loss, it sends request packets as a Possion process to its parent node p until it gets a corresponding response packet. Upon receiving the request packet, p uses the *randomcast* service to randomly decide whether to forward the packet to its parent or to one of its other children. If p decides to forward the packet to the child, its address information will be inserted into the packet so that the recipient of the request can subcast the repair packet directly the affected subtree.

A Randomized Reliable Multicast Protocol (RRMP) [ZK01] is another tree-based reliable multicast protocol aiming to improve robustness and reduce regional implosion using randomization. RRMP includes two error recovery phases, local recovery and remote recovery. when a receiver detect a packet loss, it starts local recovery and remote recovery simultaneously. In the local recovery phase, it selects another receiver in its region uniformly at random and sends a request packet to that node which replies with a repair packet if it

has the missing packet. Otherwise, the request packet is simply ignored. Once a request packet is sent, a timer is set up. When the timer expires, the receiver keeps sending request packets to other randomly chosen receivers until a repair packet is received. In the remote recovery phase, the receiver experiencing the packet loss randomly chooses a remote receiver in its parent region with a small probability. Upon receiving the repair packet from a remote receiver, the receiver will multicast the packet to all the receivers in its local region. RRMP is designed to work with the existing IP multicast framework without modification or new service provision in routers.

2.1.4 Other schemes

In a radical departure from most previous work, an Internet Reliable Multicast Architecture (IRMA) [KS99] uses a unique method for error recovery. IRMA demands no modification on the function of end hosts and supports TCP as the multicast transport protocol. The routers close to end hosts play an important role in IRMA. They are responsible for intercepting control (SYN, SYN+ACK, ACK) and data packets incurred in a TCP connection and acting as a multicast agent to communicate with other multicast group members. End hosts have no any sense about the difference between unicast and group communication through TCP. An IRMA-aware multicast router supports dynamic joins and leaves by caching the SYN packet. Each receiver replies with an ACK packet for each data packet received. The intermediate IRMA-aware routers are responsible for performing ACK aggregation and guaranteeing that only one ACK packet is received by the source. IRMA also supports local recovery by placing repair servers at the roots of multicast subtrees that experience high loss or at the boundaries of heterogeneous networks.

2.2 Congestion Control

2.2.1 Challenges for congestion control

A congestion control scheme is an indispensable part of a reliable multicast protocol. The purpose of the congestion control scheme is to guarantee that multicast flows use the network resources fairly and efficiently. The challenges faced by the congestion control scheme are listed as follows.

- **Inter-flow fairness**: When sharing network resources with other flows such as TCP flows, it should give a clear definition of fairness among different flows and implement a simple policy to realize the fairness.

- **Inter-receiver fairness**: For different receivers in the same multicast group, it stipulates that receivers experiencing different network conditions should have different treatment. For example, when two receivers suffer from a different degree of congestion on the path from the source to themselves, they should have different receiving rates. That is, a receiver should not be penalized on congestion in an unrelated part of the multicast tree.

- **Drop-to-zero problem**: It is also known as loss path multiplicity problem[SD99] in some papers. When the multicast group scales to a large number of receivers, the independent packet loss on multiple receivers will make the source react to every loss feedback and reduce the sending rate. Eventually the sending rate will shrink to zero quickly and the network resources are underutilized.

Based on the receiving rate perceived by receivers, the congestion control scheme can be classified into two categories: single-rate schemes and multi-rate schemes. In the singe-rate scheme, all the receivers have the same receiving rate. In the multi-rate scheme, the receivers may have different receiving rates.

2.2.2 Fairness issues

Since a congestion control scheme is essentially a distributed algorithm for users to share network resources. The first challenging issue is how to allocate bandwidth between TCP and multicast flows.

As for bandwidth sharing among multiple network flows, there are two main theoretical definition of fairness: max-min fairness[DR87] and proportional fairness[FK97], [FA98]. Max-min fairness stipulates that resources are allocated to network flows in order of increasing demand and no flow receives more resources than those it requested. At the same time, network flows with unsatisfied demands split the remaining resources. In other words, a max-min fair allocation guarantees that there is no way to give more bandwidth to any flow without decreasing the share to a flow of lesser or equal bandwidth. Proportional fairness allocates resources to flows by maximizing a logarithmic utility function under the capacity constraint. The resultant solution tends to penalize long flows more severely than short flows. It is difficult to construct an effective algorithm to realize the theoretical fairness criteria.

[AJ01] and [HM98] advocate that multicast flows should obtain a larger share of bandwidth than TCP flows. [AJ01] proposes a bandwidth allocation algorithm that allows routers to allocate more bandwidth to multicast flows as a reward for its bandwidth saving and an incentive for speeding up the deployment of multicast. The proposed bandwidth policy is called logarithmic receiver dependent strategy (LogRD). The LogRD policy takes into account of the number of downstream receivers when a node allocates link bandwidth to different flows. Namely, the bandwidth of a specific link allocated to a particular flow is logarithmic in the number of receivers that are downstream of that link. The strategy can achieve the best tradeoff between receiver satisfaction and fairness. However, no congestion control algorithm is given.

[HM98] puts forward a Random Listening Algorithm (RLA) to guarantee that multicast flows can get more bandwidth of a congested link than unicast flows. In the RLA, each receiver sends selective acknowledgement using the

same format as TCP SACK[MJS96] to the source. The source detects congestion by identifying the discontinuous ACK sequence number or timeout. Upon inferring congestion, the source halves its window size with probability $\frac{1}{n}$, where n is the number of receivers reporting frequent losses. Once a packet is acknowledged by all the receivers, the window size is increased by one. In this way, the RLA can achieve the essential fairness between multicast and unicast flows. At the same time. it also ensures that the ratio of multicast throughput to TCP throughput is bounded between two constant numbers so that TCP flows are never starved due to the aggressiveness of multicast flows.

Given the ubiquitous presence of TCP flows in the Internet, [SK99] insists that non-TCP flows should be TCP-friendly when they compete for bandwidth with TCP flows. A TCP-friendly flow is defined as one whose arrival rate does not exceed that of a conformant TCP flow in the same circumstance. An approach to identify unresponsive and "not TCP-friendly" flows is proposed. Most of the previous multicast congestion control schemes are designed to meet the TCP-friendly criterion.

[TM98] gives a definition of inter-receiver fairness for multicast Available Bit Rate (ABR) sessions, which an enhanced best-effort service and allows the source to learn the available capacity information and adapt to it dynamically. The inter-receiver fairness function for each receiver is given by the following formula.

$$F(r) = \frac{min(r_i, r)}{max(r_i, r)}$$

where r_i is the actual receiving rate and r is the isolated rate which is defined as the rate that would be achieved if the receiver was the only one in the multicast group and all the network conditions were unchanged. The inter-receiver fairness function for a multicast group is the sum of the individual fairness values. That is

$$IRF(r) = \sum_{i=1}^{n} \alpha_i F_i(r)$$

where α_i is a weight and their sum is equal to 1. In the proposed scheme,

the inter-receiver fairness is significant only for applications with high loss tolerance.

2.2.3 Single-rate schemes

In single-rate schemes, the source has a single sending rate and all the receivers have the same receiving rate based on the slowest receiver. A single-rate scheme can be further grouped into two classes: window-based approaches and rate-based approaches.

Window-based approaches

Window-based approaches adjust window size to regulate rate by using a TCP-like method. MTCP[LN99], AER/NCA[SS00] and PGMCC[LR00] are window-based congestion control schemes.

Multicast TCP (MTCP) [LN99] organizes the receivers into a hierarchical multicast tree for congestion control and error recovery. The internal tree nodes are referred as *sender agents* (SAs), which are responsible for aggregating and delivering hierarchical congestion reports to the source for rate regulation. Each SA keeps two variables, the size of congestion window (*cwnd*) and the estimated number of packets in transit (*twnd*), to estimate the minimum bandwidth available in the path from the source to its children. According to the feedback received from all the children, a SA can generate a congestion summary including the minimum *cwnds* (minCwnd) and maximum *twnds* (maxTwnd) and send it to its upstream SA or the source. The source can eventually adjust its sending rate based on the congestion condition inferred from the summarized congestion reports. A selective acknowledgement scheme is used to eliminate the drop-to-zero problem caused by independent loss.

Active Error Recovery and Nominee Congestion Avoidance (AER/NCA) [SS00] uses a source-based rate regulation algorithm to adjust the transmission rate according to the feedback from the worst receiver which is called the *nominee*. Each receiver dynamically estimates its end-to-end loss probability, p, and round trip time, T, from itself to the source. Then it periodically

unicasts the values p and T to its nearest upstream active router. The active router computes the product of $\sqrt{p}$ and T for each downstream receiver and then chooses the one with the highest $\sqrt{p} * T$ as the "worst" receiver. The information about the worst receiver is sent to the upstream active router, and so on and so forth. Using the same approach, the source can choose the slowest receiver from all the receivers as the nominee. The source then adjusts the congestion window by using an algorithm similar to TCP based on the feedback from the nominee. In this way, it guarantees that TCP-friendliness can be achieved.

Pragmatic General Multicast Congestion Control (PGMCC) [LR00] also dynamically chooses a representative, which is called the *acker*, from all the receivers. The source adjusts transmission rate according to the ACKs sent by the *acker*. The *acker* selection is based on the comparison of value $T^2 * p$ for each receiver, where T is the round trip time and p is the loss probability. The acker selection process need not evaluate the value $T^2 * p$ of all the receivers. This can be done on demand. For example, when a NAK from a specific node is received, the selection process can be initiated to decide whether the current acker need switch to the new node. The innovative part of PGMCC is that the migration of acker is not deemed as a *change* in the acker, but as a *move* of the acker to a different location. In this way, the measurement errors can be mitigated and assimilated and fast response to changes in the receiver set and/or network conditions can be achieved.

Golestani and Sabnani [SKG99] put forward a hierarchical approach of a window-based congestion control scheme after delving into the fairness relationship between window-based and rate-based schemes. They advocate to adopt a distinct window size for each receiver rather than using a common one for all the receivers so as to carry a sustainable throughput. However, they do not provide actual methodology and protocol details for implementation.

Rate-based approaches

Rate-based approaches regulate sending rate by adjusting the rate directly. TFMCC[MS99], REMCC[SM00], MCCR[DK98], and RACCOOM[YC01] belong to this category.

TCP Friendly Multicast Congestion Control (TFMCC) [MS99] adjusts the sending rate based on a TCP throughput equation. In TFMCC, each receiver computes its own loss fraction and sends it to the source. Loss fractions sent by different receivers are aggregated by some hierarchical mechanism. Finally the source can receive the loss fraction from the slowest receiver and adjust the sending rate according to a TCP throughput equation.

A Rate-based End-to-end Multicast Congestion Control (REMCC) [SM00] adjusts the sending rate according to the additive increase and multiplicative decrease (AIMD) principle. In REMCC, the worst receiver is dynamically chosen as an *agent*. The *agent* sends positive or negative feedback to the source every RTT. The feedback of an agent is aggregated over one RTT. Namely, if there are multiple losses in one RTT, only one negative feedback packet is sent. The source adjusts sending rate according to this feedback based on AIMD.

A Multicast Congestion Control Mechanism for reliable multicast (MCCR) [DK98] dynamically chooses a group of receivers as representatives for the whole group. Each representative sends Congestion Clear (CC) and Congestion Indication (CI) packet, which are similar to ACKs and NACKs, to the source. A Least Recently Used (LRU) algorithm, which is based on the time since the last CI was sent, is used to substitute an old representative with a new one. To avoid congesting the network, representatives send CCs randomly with the probability $\frac{1}{N}$ where N is the number of representatives. The CCs and CIs sent by representatives are also responsible for suppressing those feedback packets possibly sent by non-representative receivers before their suppression timers go off. The source uses a TCP Vegas-like[JP95] rate adaptation algorithm to adjust the rate according to the CCs and CIs sent by

the representatives.

In the RAte-based Congestion COntrOl scheme for Multicast (RACCOOM) [YC01], each receiver sends an acknowledgement packet to the source when it receives a packet. Every on-tree router is responsible for aggregating the feedback packets to avoid ACK explosion. The source also uses a TCP Vegas-like[JP95] method to adjust rate according to the aggregated ACKs.

Because the nature of single-rate multicast is to adapt to the worst receiver, almost all the schemes above need a mechanism to choose the slowest receiver. The drawback is that oscillating behavior may appear in the case that network and traffic change very dynamically. Also, single-rate schemes can not address the inter-receiver fairness issue.

2.2.4 Multi-rate schemes

In multi-rate schemes, the source or intermediate elements, for example routers or transcoders, usually have the ability to transmit the same data at different rates over multiple streams. It is up to receivers to decide which streams they want to and have the ability to listen. RLM[SV96] and LVRM[XS98] are representatives of layered approaches. However, they do not address the interaction between multicast flows and TCP flows.

Receiver-driven Layered Multicast (RLM) [SV96] is a rate-adaption protocol to address the heterogeneous network problem with the help of the layered coding technique. The source encodes the data into several layers and transmits each layer using a different multicast group. The receivers are responsible for adding and dropping layers by dynamically joining and leaving multicast groups. When congestion is perceived, the receiver drops a layer. When there is no congestion and spare capacity is detected, the receiver adds a layer. In this way, each receiver can adapt to its optimal level of subscription. In order to avoid transient congestion by join experiments of each individual receiver, a "shared learning" algorithm is proposed to coordinate the joining experiments for the receivers in the same local region. However, the protocol is designed for an environment where there is no interfering traffic.

Layered Video Multicast with Retransmissions (LVRM) [XS98] also uses the layered coding technique to distribute video over a heterogenous wide area network with best effort delivery support and an end-to-end control mechanisms. In particular, LVRM encodes the video data into three streams: I, P, and B. Each of these streams is transported using a separate multicast group. Receivers adapt to network conditions by dynamically joining and leaving those groups. A hierarchical approach is deployed to coordinate the joining and leaving actions of different receivers. Different receivers are classified into different domains according to their geographical location or administrative region. A special receiver is chosen as an agent in each domain. These agent are responsible for collecting and distributing relevant information to achieve shared learning among members in the same domain. LVRM does not address the interaction with other flows such as TCP either.

L. Vicisano et, al. [LL98] have proposed a TCP-friendly scheme for layered multicast congestion control. It is a receiver-driven congestion control algorithm. Receivers react to congestion by leaving multicast groups and dropping layers. If no losses are experienced for a period of time, the receivers subscribe to layers by joining multicast groups. Some packets in the data stream are specially flagged as *synchronization points* to coordinate the join attempts from different receivers. The source also sends probes to mitigate the bad effect of long-lasting leave delays. With appropriately spaced synchronization points and sender-initiated probes, the congestion control algorithm can achieve provably fair bandwidth sharing with TCP flows on congested links. However, it is more suitable for the real-time audio/video application.

[SJ98] is a multicast rate control scheme. It proposes a method to use multiple multicast groups to transmit data to different receivers at different rates. However, their main concern is to address the static heterogeneous problem created by the presence of a resource bottleneck. They do not cope with the congestion created by dynamic traffic changes. One drawback of the approach is that some data are transmitted several times on some links.

[DJ99] gives a theoretical analysis on the fairness between single-rate schemes

and multi-rate schemes. Compared with single-rate schemes, multi-rate schemes generally can improve the fairness properties of a multicast network. [TE99] proposes an approach to achieve inter-receiver fairness using two multicast groups for loss tolerant applications. There are two groups per multicast session, the base group and the variable group. The rate in the variable group is greater than that in the base group. When a receiver in the variable group experiences losses that exceeds its loss tolerance, it can leave the variable group and join the base group. When a receiver in the base group does not experience losses for a period of time, it may join the variable group. In this way, inter-receiver fairness can be achieved to some degree. However, when the proposed scheme competes for bandwidth with TCP flows, fairness can not be achieved without using a rate reduction factor.

2.3 Summary

In this chapter, we summarize the challenges for the reliable multicast: ACK implosion, NAK implosion, duplicate repair packets in error recovery and inter-flow fairness, inter-receiver fairness, drop-to-zero problem in congestion control. Representative approaches to tackle these challenges for error recovery are classified and presented in randomized schemes and hierarchical schemes. Those for congestion control are categorized and discussed in single-rate schemes and multi-rate schemes. In the next chapter, our approach to address those challenges will be presented in detail.

Chapter 3

Detailed Description of ARMC

In this chapter, we elaborate on the details of our protocol and discuss the design alternatives for specific mechanisms. We discuss the error recovery scheme and congestion control scheme separately.

3.1 Network Environment

The ideal network platform for our protocol is an active network which allows applications to inject specific service routines into routers to enhance the performance of particular applications. Our scheme does not require that all the routers in the Internet be active. Only a small number of active routers can make our protocol achieve considerable performance improvement. Even if the architecture of routers in the Internet does not conform to the active network principle, our protocol still can be deployed and achieve good performance if routers implement our protocol and have soft-state memory space for the protocol to use.

We also assume that routers provide subcast functionality. Subcast is the ability that a router in a multicast tree can send a packet to the subtree rooted by itself. Multicast routing protocols should make slight modifications so that feedback packets (ACKs/NAKs) sent by receivers to the source follow the exact reverse path from source to receivers. In this way, it can guarantee that active routers can reliably infer which downstream subtree has suffered from packet loss or congestion and perform exact local recovery to restrict the scope of

repair packets or conduct specific congestion control and avoidance. The whole protocol involves three logical entities: source, active routers, and receivers. The source is the entity from which data packets originate. A receiver is the entity that actually consumes the data packets sent by the source. An active router is the router that can perform some "active" processing on packets.

3.2 Protocol Overview

The Active Reliable Multicast protocol with Congestion Control (ARMC) is a one-to-many reliable multicast protocol. ARMC includes two parts: an error recovery scheme and a congestion control scheme. It is designed to achieve good performance in an active network context. It can also be deployed in the current Internet with limited support from routers. Similar to ARM[LS98] and AER/NCA[SS00], ARMC also utilizes soft-state storage in active routers to enhance the performance and scalability of the error recovery scheme and facilitate efficient congestion control. ARMC uses not only NAKs but also ACKs as feedback packets. Both active routers and receivers can check the packet sequence number gaps and send NAKs to the source. NAKs can act as both a loss indication signal and a congestion indication signal. The error recovery scheme uses a NAK suppression and aggregation mechanism in active routers to guarantee that at most one NAK for the same packet loss is forwarded to upstream active routers or the source. It also uses a local recovery and scoped retransmission mechanism to confine the impact of a packet loss in a small region and keep the number of receivers that receive the duplicate repair packets as small as possible. As a result, the average recovery time and the number of duplicate repair packets received by receivers is significantly reduced. ARMC also has an efficient hierarchical round trip time (RTT) measurement mechanism to allow the source, active routers, and receivers to quickly learn their downstream RTT or upstream RTT, which is used to set timers used in both the error recovery scheme and the congestion control scheme.

The basic idea of the congestion control scheme is that active routers use

buffers (we call them congestion smoothing buffers) to smooth the congestion by reducing the forwarding rate when congestion occurs. When congestion is gone, it increases its forwarding rate to utilize the available bandwidth. In the case that the congestion is heavy and beyond its ability to control, the active router sends a congestion indication signal to upstream active routers so that they can slow down the forwarding rate to accommodate the congestion. And so on and so forth, until the sender is informed of the congestion and eventually slows down its sending rate to avoid severe congestion. With different buffer sizes and a rate regulation algorithm, a multicast flow can obtain a different share of bandwidth of congested links with TCP flows and other multicast flows. Furthermore, the congestion that occurs on a specific link only affects the throughput of receivers below that link. Those receivers that are not located in the subtree affected by the congested link do not suffer from throughput deterioration. In this way, inter-receiver fairness can be achieved. In addition, every on-tree active router can detect the packet loss and report it to its upstream active router. This makes response time fast. Actually each active router is responsible for monitoring the links between itself and its adjacent upstream active routers and smoothing the congestion that occurs on the links within the subtree rooted by itself. When the number of active routers is up to some value and the average number of downstream links is bounded by a specific value, the number of links managed by each active routers is fairly small. Thus, the inherent feedback implosion problem and drop-to-zero problem will be automatically addressed. Scalability is then achieved. In addition, ARMC can accommodate heterogeneous network and heterogeneous receivers. However, as an initial study, although we discuss the heterogeneous cases, we only evaluate the homogeneous cases in this thesis.

3.3 The Error Recovery Scheme of ARMC

An error recovery scheme is essential for a large-scale multicast. The function of an error recovery scheme is to guarantee reliable data delivery to receivers.

3.3.1 Overview

The basic design principle of the error recovery scheme in ARMC is simplicity and effectiveness. The error recovery scheme has the following characteristics.

- **Early packet loss detection**: Besides receivers, active routers also detect packet sequence number gaps and send NAKs to upstream nodes. This is similar to AER/NCA. However, in ARMC, once it detects a packet loss, an active router immediately sends a NAK to upstream nodes and starts an error recovery process. In AER/NCA, the active router can send a NAK only when its suppression timer expires.

- **NAK suppression**: When a data packet is lost on some intermediate link, it is possible that multiple active routers and receivers will detect the packet loss and send the NAK packet to initiate the error recovery process. To reduce the number of NAKs sent by active router and receivers, a NAK suppression mechanism is necessary. In ARMC, each time when an active router detects packet loss, it sends a FAKE_DATA packet to downstream nodes to suppress NAKs that would be sent by downstream active routers or receivers.

- **NAK aggregation**: It is possible that a packet is lost on several different paths and several NAKs for the same lost packet arrive at an active router. A NAK aggregation mechanism is indispensable to aggregate those NAKs for the same lost packets and guarantee that only one NAK is forward to the upstream active router or the source. ARMC uses NAK records in active routers to aggregate NAKs with the same packet sequence number. This is similar to ARM.

- **Local recovery**: Local recovery is used to accelerate the error recovery process and reduce the error recovery load of the source. In ARMC, active routers are responsible for local recovery so that most lost packets are retransmitted by the nearby active routers. Each active router utilizes the packets in its soft-state storage to realize local recovery.

- **Scoped retransmission**: Scoped retransmission is used to reduce the number of duplicate repair packet received by receivers. In ARMC, when an active router sends a repair packet, it sends it only to the downstream links from which NAKs come according to the NAK record information. In this way, the undesirable impact of duplicate repair packets is confined to a limited scope. This is similar to ARM[LS98].

- **Hierarchical RTT measurement**: ARMC proposes a hierarchical RTT measurement mechanism to obtain the RTT information used to set the timer for buffered packets in active routers and the retransmission timer in receivers. This hierarchical RTT measurement mechanism is initiated by upstream active routers or the source. Although AER/NCA[SS00] also has a hierarchical RTT measurement mechanism, it is initiated by downstream receivers or active routers.

- **Periodical ACK mechanism**: As stated in [BJ98], a pure-NAK based error recovery scheme can lead to an infinite buffer problem in order to avoid deadlocks in the source. In our protocol, receivers can periodically send delayed ACKs to remove this problem. The ACKs can also remove data packets in soft-state buffers before timeout and release the buffer space earlier.

These mechanisms are described in more detail in the next section. Figure 3.1 illustrates the basic algorithm for our error recovery scheme.

In order to facilitate exposition of our protocol, we define several terms below.

- **Adjacent active routers**: Two active routers are adjacent if there are no other active routers on the path between them. There may exist other non-active routers on the path between them.

- **Intermediate links**: Links between two adjacent active routers or between active routers and the source. The routers between two adjacent active routers are not active.

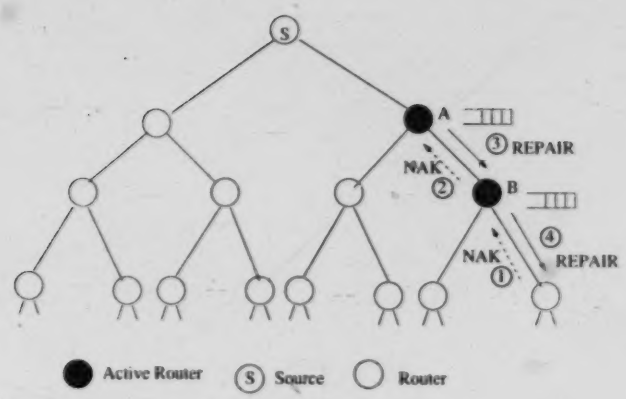
- **End links**: Links between the last active routers and receivers. There may be non-active routers between active routers and receivers.
- **Resent NAKs**: NAKs sent by receivers due to retransmission timeout.
- **NAK record**: A data structure in active routers used to record the information of received NAKs. It includes the information of packet sequence number and downstream links from which the NAKs come.
- **Repair record**: A data structure in active routers used to record the information of repair packets sent. It includes the information of packet sequence number and downstream links to which repair packets are sent.
- **Upstream RTT**: RTT from an active router or receiver to the source. This is used by active routers or receivers to set timers for NAK records.
- **Downstream RTT**: RTT from the source or an active router to the receivers. The largest downstream RTT is the time from the source or active routers to the farthest downstream receivers. It is used by active routers to set timers for repair records.

3.3.2 Early packet loss detection

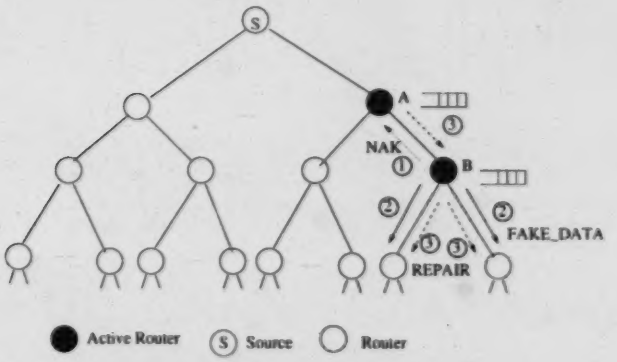
When there are enough active routers, packet loss mostly occurs on the intermediate links rather than on the end links. Therefore, the mechanism that only receivers can detect packet sequence number gaps and send NAKs is not the fastest way to do error recovery. In our protocol, active routers can also detect packet sequence number gaps and send NAKs to upstream nodes. In the case that repair packets are lost, only receivers can send NAKs again. Additionally, this mechanism can also facilitate congestion detection in the congestion control scheme.

3.3.3 NAK suppression and aggregation

The side effect of an early packet loss detection mechanism is that one lost packet may incur multiple NAKs. If a packet is lost on the intermediate links,



(a) **NAKs originated from receivers.** (1) active router B receives a NAK from a downstream receiver. (2) when B finds that it is the first NAK with a different packet sequence number, B forwards the NAK to the upstream active router A. (3) A has the requested repair packet, it sends the repair packet only to the downstream link from which the NAK comes. (4) when B receives the repair packet, it buffers that packet and forwards the repair packet only to the link from which NAKs come according to the subscription information in the NAK record.



(b) **NAKs originated from active routers.** (1) when active router B detects a packet sequence number gap, it sends a NAK to upstream nodes. (2) at the same time, B subcasts a FAKE_DATA packet to the subtree to suppress the NAKs generated by downstream active routers or receivers. (3) A has the requested repair packet, it sends the repair packet only to the downstream link from which the NAK comes. When B receives the repair packet, it subcasts the repair packet to the whole subtree.

Figure 3.1: Error Recovery Scheme.

the first downstream active router would detect the packet loss and send NAKs to upstream nodes. At the same time, other downstream active routers and receivers would also detect that packet loss and send NAKs to upstream nodes. All of them should be suppressed by the upstream active routers. These useless NAKs will consume the network bandwidth and increase the processing load of active routers.

To avoid the above problem, the NAKs are suppressed by the following means. Each time when active routers detect packet loss and send NAKs to upstream nodes, they also subcast a FAKE_DATA packet to all the downstream nodes. When downstream active routers or receivers receive FAKE_DATA packets, they simply create a NAK record to facilitate error recovery and forward it to downstream nodes if possible. In this way, only the active router that is first aware of packet loss sends NAKs. The number of NAKs sent by downstream active routers and receivers is significantly reduced. The mechanism can suppress only the NAKs originated from packet loss on intermediate links. It does not suppress lost NAKs on end links or resent NAKs. This is reasonable since the packet loss on the end links would not create cascading NAKs. We also assume that the number of resent NAKs is small and they also can be effectively handled by the NAK aggregation mechanism in active routers. Thus it is not necessary to design a special mechanism such as a suppression timer to suppress NAKs originated from receivers and resent NAKs. Avoiding suppression timer simplifies the protocol.

Active routers also aggregate the NAKs with the same packet sequence number. Each time when an active router receives a FAKE_DATA packet from an upstream node, it creates a NAK record. In addition, a NAK record is also created when a NAK with a distinct packet number arrives from a downstream node and no repair packet is available in the active router. NAK records are used to suppress the subsequent NAKs. Each NAK record is also associated with a timer and is removed if no repair packet comes before the timer expires. The timer value to cache a NAK record is at least one upstream RTT, which is the time expected to receive a repair packet. Each time when

a repair packet is sent, a REPAIR record is created and used to suppress the possible NAKs sent by receivers before they receive the repair packet. The timer value to cache a REPAIR record is at least one largest downstream RTT, which is the time when the receivers should have received the repair packets.

3.3.4 Local recovery and scoped retransmission

Local recovery is indispensable in order to achieve fast error recovery. In ARMC, active routers buffer received data packets or repair packets for an amount of time (usually one largest downstream RTT) to implement local recovery. When a NAK comes before the data packets are removed due to timeout, a repair packet can be sent immediately to the downstream links for error recovery. Thus, usually receivers directly obtain repair packets from intermediate active routers and thus the recovery latency is greatly shorten.

It is important to limit the scope of repair packets because duplicate and useless repair packets waste precious network resources such as bandwidth. ARMC can effectively solve this problem by sending repair packets only to the downstream links from which NAKs come according to the subscription information in the NAK records. In this way, the impact of a packet loss is confined to a limited scope. This local recovery and scoped retransmission mechanism is similar to ARM.

3.3.5 Hierarchical RTT measurement

The aim of this mechanism is to let the source and active routers learn the upstream RTT and largest downstream RTT and let receivers learn the upstream RTT. As stated in previous sections, ARMC needs to set a timer to manage the buffered data packets, NAK records, and REPAIR records. Each active router maintains the upstream RTT, the downstream RTT, and the source RTT which is the largest downstream RTT of the source. The upstream RTT is used to set the timer for the NAK records. The downstream RTT is used to set the timer for the REPAIR records and data packets. The source RTT

is used to facilitate the computation of the upstream RTT and the downstream RTT. In addition, the congestion control scheme discussed in the next section also needs the RTT information to infer congestion. With the hierarchical RTT measurement mechanism, the source and active routers can quickly learn the upstream RTT and downstream RTT by querying the information directly from the next downstream active routers rather than from receivers. Figure 3.2 illustrates our hierarchical RTT measurement mechanism.

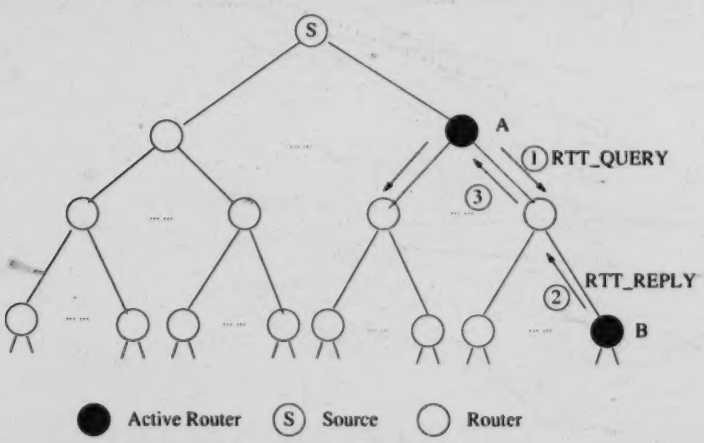


Figure 3.2: Hierarchical RTT measurement. (1) active router A sends RTT_QUERY packet to downstream nodes. (2) active router B intercepts RTT_QUERY packet and sends back a RTT_REPLY packet without forwarding it further to downstream nodes and computes its upstream RTT. (3) when receiving RTT_REPLY packets, A computes its largest downstream RTT.

The mechanism is initiated by the source or active routers. In order to measure the RTT, we design two new packet types, RTT_QUERY packet and RTT_REPLY packet. The source and each active router independently measures its own downstream RTT by periodically sending a RTT_QUERY packet to the downstream active routers or receivers. The RTT_QUERY packet includes the information of upstream RTT of the active router and the largest downstream RTT of the source. When a downstream active router or receiver receives the RTT_QUERY packet, it first saves the largest downstream RTT of the source and sends back a RTT_REPLY packet. The RTT_REPLY packet

includes the timestamp of the RTT_QUERY packet and the downstream RTT of the active router that replies to the RTT_QUERY packet. Then the active router computes its rough upstream RTT by subtracting its own downstream RTT from this source RTT.

Although the upstream RTT computed by this simple mechanism overestimates the RTT, especially for the nodes close to the source, the effect may not be serious since the upstream RTT is used to set timeout values of NAK records most of which are removed by repair packets. The upstream RTT of a receiver is the source RTT in this mechanism. In the case that repair packets are lost, it will take a bit longer for receivers to resend NAKs which will make the recovery time somewhat longer. However, a longer timeout value for the NAK retransmission timer also has an advantage that receivers can withstand the wrong estimation of RTT and thus reduce the number of resent NAKs in the case that no repair packets are lost.

Each time the source or active routers send a RTT_QUERY packet, they set up a timer to collect the RTT_REPLY packets. The timer should be at least one current measured largest downstream RTT. When these timers go off, they first compute the largest downstream RTT of all the received RTT_REPLY packets by adding the measured RTT from itself to adjacent downstream active routers or receivers and the downstream RTT of that active router that sends back the RTT_REPLY packet. Let *crtt* represent the measured largest downstream RTT. Then they use a TCP-like mechanism to compute the smoothed largest downstream RTT (*srtt*) and smoothed RTT variance (*rttvar*). If it is the first measurement, the *srtt* is set to *crtt* and the *rttvar* is set to $\frac{1}{2}crtt$. Otherwise, the *srtt* is computed by $\frac{7}{8}srtt + \frac{1}{8}crtt$ and *rttvar* is calculated by $\frac{3}{4}rttvar + \frac{1}{4}|srtt - crtt|$. The smoothed RTT and variance will be used to set the timer to buffer data packets and REPAIR records. The timer to keep data packets and REPAIR records in an active router is computed using an algorithm similar to that in TCP to compute the timeout value. Namely, the timer value is computed by $srtt + 4rttvar$. We tend to keep the data and REPAIR records a little longer than the real RTT. In this way,

we can avoid the unnecessary removal of data and increase the error recovery time in the case that the real RTT is slightly larger than the estimated one.

3.3.6 Periodical ACK mechanism

In a pure-NAK based scheme, the source has no way to decide when to remove the retransmission buffer. Thus it will create an infinite buffer problem and may use up memory resources of the source. This is the motivation for our periodical ACK mechanism. Receivers periodically send selective ACKs, which acknowledge all the previous received consecutive packets, to upstream active routers or the source. Active routers aggregate the ACKs sent by all the downstream receivers and send the *minimum* ACK of all the received sequence numbers to upstream active routers or the source periodically. The period for sending an ACK is usually one largest downstream RTT of the source. The ACKs can also remove data packets and NAK or REPAIR records in active routers before timeout and release the buffer space earlier. [YY00] has a similar ACK mechanism.

3.3.7 Buffer management strategy

It is important to efficiently manage the soft-state buffers in active routers. We store data packets, NAK records, and REPAIR records in 3 separate buffers. For each of the buffers, we use a single timer to manage all the packets in the buffer. We keep track of the packets that enter the buffer since the last timeout and remove them when the next timeout occurs. In this way, the average buffered time for a packet is a little longer than the timeout value. But the management mechanism is simple.

3.3.8 Scalability issues

The NAK implosion at the source is the main barrier to achieve scalability for the receiver-initiated reliable multicast protocol. With active routers that retransmit most of the lost packets, our scheme can effectively solve the NAK implosion problem. As will be seen in the next chapter, by using only a small

number of active routers for each multicast tree, good performance can be achieved. Therefore, there is less chance that a large number of multicast groups will use the same active routers. In addition, the memory requirement of each multicast group is very small. This makes it possible for our scheme to support multiple multicast groups without creating new scalability problems.

3.4 Congestion Control

A congestion control and avoidance scheme is a fundamental part of a multicast protocol to support large-scale group-based applications. The function of a congestion control scheme is to facilitate efficient and fair share of network resources.

3.4.1 Overview

Our congestion control scheme is a router-assisted window-based hierarchical mechanism. The idea is simple. When congestion occurs in some downstream links, active routers slow down the data forwarding rate and use congestion smoothing buffers to store the packets created due to the difference between packet arrival rate and forwarding rate. Only when the congestion is severe and beyond the ability of an active router to accommodate, does it inform the upstream active router or source to slow down and react to congestion. Our scheme tries to let active routers accommodate most of light and medium congestion conditions and let the source react only when severe congestion conditions occur. This is validated by the observation[MJ96], [MH97] that most of the losses in MBONE occurs near the receiver and the majority of burst loss last a short period of time. Though our congestion control scheme is designed for reliable multicast applications, it can also be used for non-reliable multicast applications. Fig 3.3 illustrates our congestion control scheme.

The congestion control scheme includes the following mechanisms.

- **Congestion detection**: Active routers or receivers detect congestion by checking for a packet sequence number gap. Upon finding a packet

sequence number gap, an active router or receiver sends a NAK packet to the nearest upstream active router or the source. When receiving the NAK packet, an active router or the source can infer the congestion and set up a congestion smoothing buffer to accommodate the congestion. When the number of packets in the congestion smoothing buffer is greater than a threshold, the active router sends a CIS packet, which acts as a congestion indication signal, to the upstream active routers or the source.

- **Hierarchical RTT measurement**: The congestion control scheme also uses the hierarchical RTT measurement mechanism described in Section 3.3.5 to let active routers or the source learn the largest downstream RTT information, which is the round trip time from the source or active router to its furthest downstream receiver. The measured RTT information is mainly used by the source and active routers to adjust the sending rate or forwarding rate and infer congestion.

- **Congestion window regulation algorithm**: Each active router keeps the congestion window size of its adjacent upstream active router or the source to estimate the incoming data rate when congestion is inferred. The source or active routers adjust their congestion window using the additive increase and multiplicative decrease mechanism similar to TCP[VJ88].

- **Management of congestion smoothing buffer**: Upon detecting congestion on some downstream links, active routers initialize a buffer size using the formula $\alpha^2 + \alpha w_0$, where α is the estimated congestion duration in units of the RTT of its adjacent upstream active router, and w_0 is the congestion window size of its adjacent active router or the source at the time when congestion is detected. This is derived in more detail later.

3.4.2 Design objectives

The basic objectives of our congestion control scheme are presented as follows.

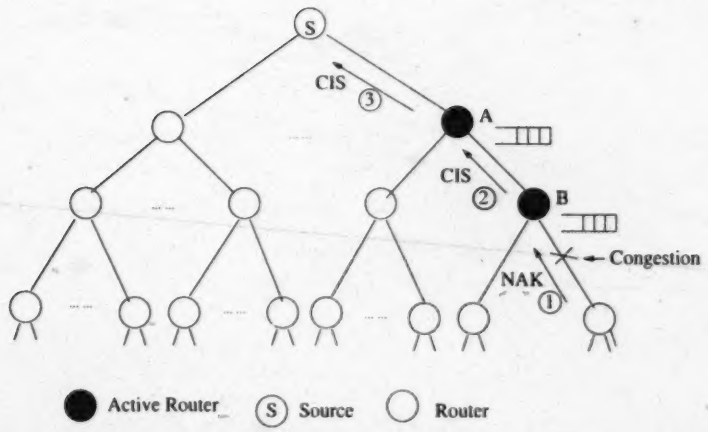


Figure 3.3: Congestion control and avoidance example. (1)active router B infers congestion from NAKs sent by downstream nodes which detect a packet sequence number gap and sets up a buffer to smooth it. (2) active router B has no ability to accommodate congestion and then sends a CIS to upstream active router A. After A infers congestion from the CIS, A sets up the buffer and reduces the forwarding rate to smooth congestion. (3)When congestion is beyond the ability of A to accommodate, A sends a CIS to the source. After the source receives the CIS, it decreases its sending rate to avoid congestion.

Adjustable share of bandwidth of congestion link: In the ideal form, we hope that applications can obtain a different share of bandwidth of a congested link by buying different amount of resources such as buffer capacity. For short-term concerns, a multicast flow can get more bandwidth than a TCP flow but it can not starve the TCP flow due to the predominant presence of TCP flows in the current Internet. One important thing to note is that the congestion control scheme can achieve "TCP-friendliness" by adjusting its parameters. In ARMC, the source and active routers use a congestion window regulation mechanism that is similar to TCP. The variation of adding congestion smoothing buffers only impacts the reaction time of the source. In the extreme case where an active router has infinite buffers for a multicast flow to use, ARMC behaves like a multiple unicast relay mechanism where each active router acts as a relay node.

Inter-receiver fairness: The previous single-rate based congestion control schemes do not address inter-receiver fairness. In single-rate based schemes, the source adapts to the slowest receiver and all the receivers have the same throughput. However, congestion usually happens only on some links. A single-rate scheme can only satisfy the requirement of the slowest receiver. It is unfair for non-congested receivers to be subject to a low throughput just because of the unrelated congestion occurring on the paths of other receivers. In our scheme, each active router controls a limited number of links and only reacts to the congestion that occurs on those links. As a result, when congestion occurs, only the receivers located in the subtree affected by the congested link would suffer decreasing throughput. Receivers that are not located in the affected subtree would not be subject to decreasing throughput until the congestion is very heavy and eventually is noticed by the upstream active routers or the source. The concept of inter-receiver fairness is a little different from the one in [TM98], [TE99] since the definition there mainly aims at the unfairness created by differences of static link capacity. Our solution mainly addresses the unfairness created by dynamic congestion on different links although our scheme can also be extended to address unfairness due to different

link capacities.

Scalability: Scalability is a main issue that all the multicast protocols should address because multicast-based applications may include a large number of receivers and involve a heterogeneous environment with various link bandwidths and delays. The main resulting problems faced by multicast congestion control are the feedback implosion problem and drop-to-zero problem. In our scheme, the feedback implosion problem is solved by appropriate feedback aggregation at active routers and the mechanism that the source and each active router only react to feedback from a limited number of adjacent downstream active routers or receivers. The drop-to-zero problem is caused by the independent packet loss on multiple receivers and source reaction to every loss feedback sent by each receiver. Our solution handles this problem by limiting the number of nodes, whose feedback the source and active routers would react to, and by decreasing the rate at most once in one RTT.

Responsiveness: Responsiveness refers to how fast a congestion control scheme can detect and react to congestion. The typical response time of TCP is one RTT or one retransmission timeout. The existing multicast congestion control schemes usually take a longer time to react due to the delay to process the feedback and choose the slowest receiver. The unresponsiveness of a multicast flow may potentially create fluctuating link conditions and potentially could endanger the stability of the network. The slow responsiveness of a multicast flow would result in unfairness with TCP flows at least in a short period of time. Our approach can effectively solve this problem since active routers can detect the congestion quickly and inform the upstream active router or source to react when congestion occurs.

3.4.3 Congestion detection

Active routers and receivers keep track of the sequence number of the last packet received. Using the last packet sequence number, active routers and receivers can detect whether there is a gap between the current packet sequence number and the last packet sequence number. Upon detecting a packet se-

quence number gap, active routers and receivers infer that congestion has happened on one or more upstream links and then they send a NAK to the nearest upstream active router or the source to react to the congestion. In ARMC, NAKs can have two roles: loss indication signal and congestion indication signal. The expected response to the former is to retransmit the missing packet, while the expected response to the latter is to reduce the forwarding rate. The first NAK sent by active routers or receivers acts as both a loss indication signal and a congestion indication signal. However, when it arrives at an active router and is handled by a congestion control agent in that active router, it acts only as a loss indication signal if it is forwarded to upstream nodes. A flag in the NAK header is used to differentiate those two roles. The justification is that congestion has been controlled by the nearest upstream active router and other upstream active routers need not react again to that congestion. Also, a resent NAK acts only as a loss indication signal. The reason is that the packet represented by the resent NAK may have been lost again and the congestion inferred by the packet loss event has been reacted to by active routers or the source. It is possible that congestion has gone. Thus, the second congestion reaction is unnecessary. The drawback of inferring congestion by NAKs originated from packet sequence number gaps is that out-of-order packets could lead to an incorrect congestion indication. Here we assume that the number of out-of-order packets are few because the data packets always follow the multicast forwarding tree and only minimal changes should be made to the tree.

When the number of packets in the congestion smoothing buffer is greater than a threshold value, *thresh_idr*, active routers send a CIS packet as a congestion indication signal to the upstream active router. In one RTT, active routers send at most one NAK or CIS as a congestion indication signal to upstream nodes. In this way, it can inform the upstream nodes of congestion while reducing the bandwidth consumed by redundant NAKs or CISes. Source and active routers infer congestion by any of received NAKs and CISes. In one RTT, the source or active routers react to congestion and reduce the

congestion window at most once.

3.4.4 Start and termination of congestion smoothing phase

When an active router infers congestion that occurs on some downstream links from a congestion indication signal (NAK or CIS), it enters a congestion smoothing phase by setting up a congestion smoothing buffer. If there is no space for creating the congestion smoothing buffer, the congestion indication signal (NAK or CIS) is just forwarded to upstream nodes.

The source keeps 2 parameters: congestion window (*cwnd*) and slow start threshold (*ssthresh*), for the sending rate regulation. Similarly, every active router also keeps 2 parameters: congestion window (*cwnd*) and congestion window of the nearest upstream active router or the source (*ucwnd*), to adjust the forwarding rate and decide whether or not to terminate the congestion smooth phase. When the source sends packets, it puts its own *cwnd* into packets. Let *pcwnd* denote the congestion window carried by a packet. Upon receiving a data packet, an active router uses *pcwnd* to update its *ucwnd* and replaces *pcwnd* with its current *cwnd* if it is in the congestion smooth phase. If an active router is not in the congestion smooth phase, it just uses *pcwnd* in the packet to update *ucwnd* and forwards the *pcwnd* as is to downstream nodes immediately. In this way, each active router can learn the congestion window size of its adjacent upstream active router. When an active router is in the congestion smooth phase, it sends a congestion window of data every RTT. The congestion smooth phase is terminated if its congestion smooth buffer is empty.

3.4.5 Congestion window regulation algorithm

When an active router enters the congestion smoothing phase, *cwnd* is initialized to *ucwnd*. During the congestion smoothing phase, *cwnd* is halved if at least one congestion indication signal (NAKs or CISes) is received in one RTT. If no congestion indication signal is received in one RTT, *cwnd* is increased by

one. In detail, the congestion window regulation algorithm for active routers is illustrated in Fig. 3.4.

- on start, $cwnd = ucwnd$;
- on congestion, $cwnd = \frac{1}{2} cwnd$;
- on no congestion, $cwnd = cwnd + 1$;

Figure 3.4: Rate Regulation Algorithm at an Active Router

The source also uses the additive increase and multiplicative decrease (AIMD) principle to adjust rate. If congestion is inferred, *cwnd* is halved and *ssthresh* is set to *cwnd*. If no congestion is inferred in one RTT, *cwnd* is increased by one if *cwnd* is greater than or equal to *ssthresh*. Otherwise, if *cwnd* is less than *ssthresh*, *cwnd* is doubled. The start-up behavior of the source is similar to TCP. It also uses the slow start mechanism to detect available bandwidth in the network. The initial value of *cwnd* is set to 1 and the initial value of *ssthresh* is set to a constant value such as 64. The congestion window regulation algorithm in the slow start phase guarantees that the source increases its rate exponentially. In detail, the congestion window regulation algorithm for the source is illustrated in Fig. 3.5.

- on initialization,
  $cwnd = 1$, $ssthresh =$ some constant value;
- on congestion, $cwnd = \frac{1}{2} cwnd$, $ssthresh = cwnd$;
- on no congestion,

$$cwnd = \begin{cases} cwnd + 1 & \text{if } cwnd \geq ssthresh; \\ 2 * cwnd & \text{if } cwnd < ssthresh. \end{cases}$$

Figure 3.5: Rate Regulation Algorithm at the Source

When the above algorithm is implemented in the real world, the congestion window of an active router and the source should be bounded by a minimum

window (MINWND) and a maximum window (MAXWND).

$$MINWND \leq cwnd \leq MAXWND$$

Note that applications can negotiate the parameters and congestion window regulation algorithm with network providers in the future. In order to be compatible with TCP and not to starve TCP flows, currently ARMC uses the above algorithm to adjust the congestion window. It is left as a future research direction to study how different congestion window regulation algorithms can affect the share of bandwidth.

3.4.6 Congestion smoothing buffer management

The congestion smoothing buffer is an essential part for the proposed congestion control scheme. Once the congestion is inferred, active routers dynamically estimate the buffer size according to the congestion window of its upstream active router (*ucwnd*) and the expected congestion period. One parameter, *thresh_idr*, is used to monitor the congestion smoothing buffer. During the congestion smoothing phase, every packet received is first put into the congestion smoothing buffer before it is forwarded to downstream nodes if the number of packets in the congestion smoothing buffer is smaller than *thresh_idr*. When the number of packets in the congestion smoothing buffer is greater than *thresh_idr*, the incoming packets are also put into the congestion smoothing buffer and will trigger a CIS packet to be sent to upstream nodes. When congestion is inferred to have gone, the congestion smoothing buffer is freed and every incoming packet is immediately forwarded to downstream nodes without extra buffering.

In the following paragraphs, we analyze the relation between the buffer size and the estimated congestion duration in order to automatically control the buffer size. Our analysis is based on a simple topology illustrated in Fig. 3.6. In the topology, there are 2 active routers, A and B. Active router A is the upstream active router of the active router B.

When congestion occurs on one of B's downstream links, B sets up a con-

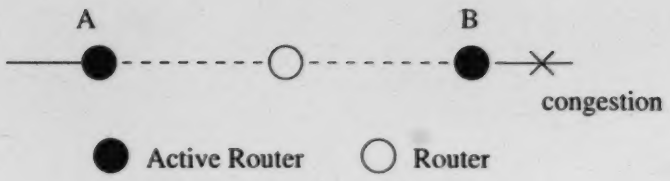


Figure 3.6: Analysis Topology

gestion smoothing buffer to alleviate the congestion. We model the congestion smoothing duration as two phases, a rate reduction phase and a rate increase phase. As illustrated in the upper graph of Fig. 3.7, the phase from t_0 to t_1 is the rate reduction phase and the phase from t_1 to t_3 is the rate increase phase. From the viewpoint of buffer usage, it can also be separated into two phases, a buffer increasing phase and a buffer decreasing phase. In the lower graph of Fig 3.7, the phase from time t_0 to time t_2 is the buffer increasing phase and the phase from t_2 to t_3 is the buffer decreasing phase.

In the analysis, let $\bar{R}_A$ and $\bar{R}_B$ denote, respectively, the average forwarding rate of the active router A and B from t_0 to t_2. Let RTT_A and RTT_B denote, respectively, the round trip time of the active router A and B. To simplify the analysis, we assume that RTT_A and RTT_B is unchanged during the congestion smooth duration. To accommodate the congestion without informing the active router A, the buffer size at the active router B can be calculated as follows.

$$buf = (\bar{R}_A - \bar{R}_B)(t_2 - t_0) \tag{3.1}$$

Obviously, we have the following relations from Fig. 3.7.

$$\bar{R}_A < R_{A_{t_2}} \tag{3.2}$$

$$\bar{R}_B > R_{B_{t_1}} \tag{3.3}$$

where $R_{A_{t_2}}$ is the sending rate of A at time t_2 and $R_{B_{t_1}}$ is the sending rate of B at time t_1.

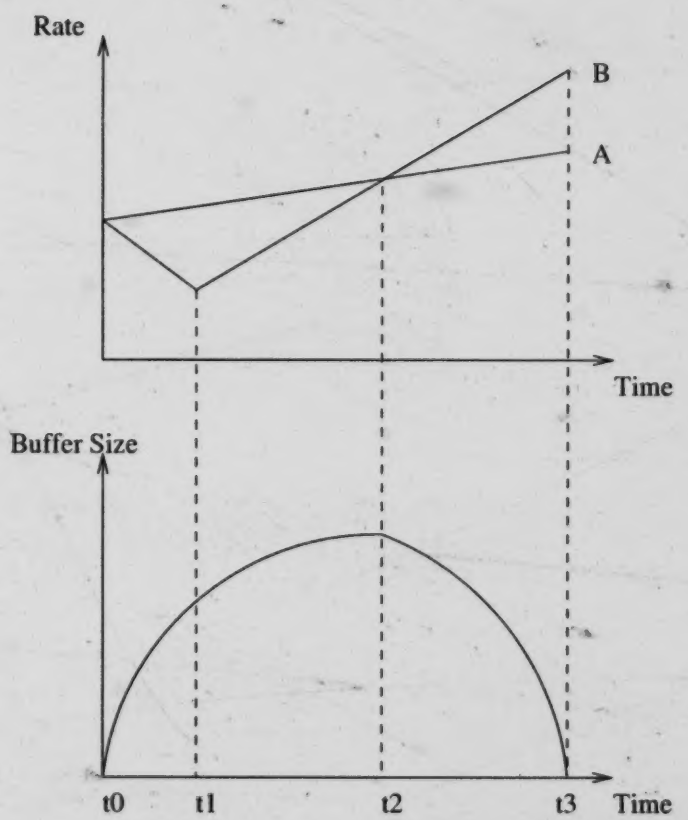


Figure 3.7: Rate and Buffer Change during the Congestion Smooth Duration

Combining the above three formulas, we have

$$buf < (R_{A_{t_2}} - R_{B_{t_1}})(t_2 - t_0)) \tag{3.4}$$

Let the congestion window at time t_0 be w_0. Also we define the following relations.

$$t_2 - t_0 = \alpha RTT_A \tag{3.5}$$

$$t_1 - t_0 = \phi RTT_B \tag{3.6}$$

where α and ϕ are positive integers. α can be deemed as the estimated congestion duration in units of RTTs of the active router.

According to the congestion window regulation algorithm in Fig. 3.4 and Fig. 3.5, the congestion window of A is $w_0 + \alpha$ at time t_2 and that of B is $\frac{w_0}{2^\phi}$ at time t_1. Then we can get the following equations.

$$R_{A_{t_2}} = \frac{w_0 + \alpha}{RTT_A} \tag{3.7}$$

$$R_{B_{t_1}} = \frac{w_0}{2^\phi RTT_B} \tag{3.8}$$

Then,

$$buf < (\frac{w_0 + \alpha}{RTT_A} - \frac{w_0}{2^\phi RTT_B})\alpha RTT_A \tag{3.9}$$

$$< (\frac{w_0 + \alpha}{RTT_A} - \frac{w_0}{2^\phi RTT_A})\alpha RTT_A \tag{3.10}$$

$$= \alpha^2 + \alpha w_0(1 - \frac{1}{2^\phi}) \tag{3.11}$$

$$< \alpha^2 + \alpha w_0 \tag{3.12}$$

The above buffer size is a conservative one. The *thresh_idr* can be computed by the following formula.

$$thresh_idr = \rho * buf \tag{3.13}$$

$$= \rho(\alpha^2 + \alpha w_0) \tag{3.14}$$

where $0 < \rho < 1$.

In the case that the buffer size is 0, an active router simply aggregates the NAKs received and sends one CIS to upstream active routers or the source

when it receives a NAK. Note that the the buffer actually occupied by a multicast flow is possibly slightly greater than the negotiated one if ρ is not appropriately chosen. The guideline to choose ρ is that it should guarantee that there are still at least $w_0 + \alpha$ free buffers when a CIS is sent since it guarantees that the ensuing packets will not overflow the congestion smoothing buffer before the upstream active router receives that CIS and slows down its forwarding rate.

3.4.7 Handling drop-to-zero problem

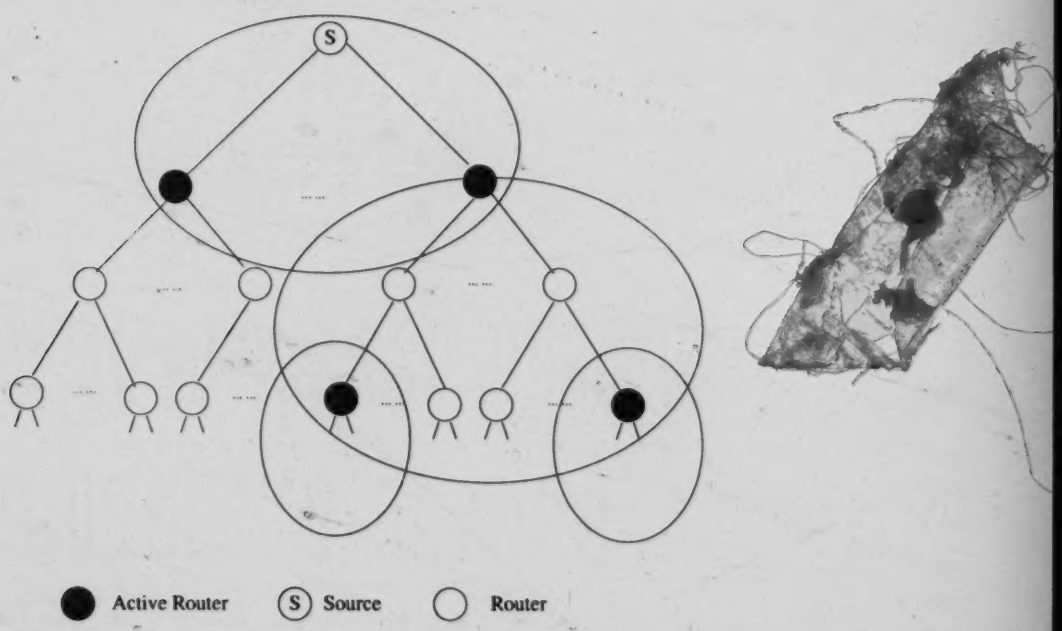


Figure 3.8: Solution to drop-to-zero problem. The multicast tree is separated into multiple subtrees managed by different active routers or the source.

The Drop-to-zero problem is also known as the loss path multiplicity problem [SD99] in the literature. It is inherent in the multicast protocols and is caused by independent packet loss on multiple paths. We handle this problem by using three approaches: (1) we split the multicast tree into multiple subtrees so that the source and each active router are only responsible for losses in

a limited number of links. (2) An active router sends at most one NAK or CIS in one RTT. (3) The source and active routers react to NAKs or CISes at most once in one RTT. Basically our scheme can handle all the parallel losses, which occur in different downstream links within the same subtree during the same RTT since all the parallel losses can only cause the source or active routers to reduce the congestion window once. Our scheme can also prevent serial losses, which occur in different links during different RTT periods since the losses hereafter in another link will be avoided by the rate regulation reaction in the source or active routers caused by the NAKs generated in the previous RTT.

3.4.8 Handling heterogeneous link bandwidths and receivers

Though we assume in this thesis that all the links have the same capacity and receivers have the same processing ability and receiving window size, this is not true in the real world. Nevertheless, our scheme can easily be extended to handle heterogeneous links and receivers. Heterogeneous links are easy to handle if the upstream links have less bandwidth than the downstream links since the downstream links can always transmit the packets without causing congestion and any packet loss. In the case that the upstream links have more bandwidth than the downstream links, active routers need to set up large enough soft-state storage to accommodate the rate difference. Surely in this case it needs a good active router architecture so that it can easily manage a large amount of storage within the network. Essentially this is a kind of relaying mechanism.

In order to avoid overrunning the receivers due to a large congestion window in the source or active routers, the receivers can include their receiving window size in the ACK. Intermediate active routers can aggregate the receiving window size so that the maximal one is sent to the upstream active router. And so on and so forth, until finally the maximal receiving window size can be learned by the source. The source can control its congestion window so that it is never greater than the maximal receiving window size. However,

even if there exists the above mechanism, it still can not guarantee that all the receivers can not be overrun since the active router adjacent to the receivers can still have a greater congestion window than the receiving window of some of its downstream receivers. Therefore, there must be a new mechanism so that the active routers can be aware of whether there are receivers that are directly adjacent to them and make their congestion window less than the minimal receiving window of all the adjacent downstream receivers. This mechanism can easily be designed. For example, we can add a flag to the ACK packet header to differentiate the ACKs directly sent by receivers and the ACKs aggregated and sent by active routers. If an active router receives ACKs directly sent by receivers, it can infer that there exists directly adjacent receivers and then keep a variable to remember the minimal receiving window of all the adjacent downstream receivers when it aggregates the ACKs. It can make its congestion window always less than the minimal receiving window of all the adjacent downstream receivers. The above mechanism is basically a prevention mechanism. Another simple solution is that receivers simply send NAKs when they are overrun. With this approach, some packets have to be resent and a portion of bandwidth is wasted. This mechanism basically favors fast receivers.

3.4.9 Possible limitations

ARMC will have problems under certain circumstances. For example, ARMC will have problems if the number of out-of-order packets are large. In this case, our congestion detection mechanism will mistakenly infer congestion from out-of-order packets by detecting packet sequence number gaps. In this paper, we assume that all the packet losses are caused by congestion. However, if there are random losses caused by packet corruption or other reasons, our scheme would interpret this kind of loss as congestion. Another possible problem is that the source adjusts congestion window size and sends new data to the network only when the sending timer goes off at the end of one RTT and it guesses that the previous window of data have left the network. These "smart

guesses" may cause problems if the measured RTT is not accurate.

3.5 Relationship between Error Recovery and Congestion Control

Basically, the error recovery scheme and the congestion control scheme are independent although they both use the estimated RTT obtained by the same RTT measurement mechanism and react to arriving NAKs with different behavior. Therefore, the error recovery scheme and the congestion control scheme can be separated on different active routers. When they are put on different active routers, they both need to deploy the RTT measurement mechanism. In addition, the congestion control scheme is only responsible for detecting packet sequence number gaps and reacting to the first NAK originated from the same packet loss and forwarding every NAK to upstream nodes. The error recovery scheme is responsible for aggregating NAKs and sending repair packets. The concrete issues originated from separating error recovery and congestion control on different active routers will be our future research work.

3.6 Handling Dynamic Membership Changes

In this thesis, we assume that the multicast tree is static and no receivers dynamically join or leave a multicast group. However, in the real world, it is possible for receivers to join or leave a group dynamically. The dynamic membership may pose a challenge to the exactness of the measured RTT since it may not be precise to use smoothed RTT estimation formula to compute the new RTT. The inaccuracy of the estimated RTT could cause problems to both error recovery and congestion control. In order to address this problem, one possible solution is to let the underlying routing protocol to notify the joining and leaving event to the reliable multicast protocol so that it can avoid using the last estimated smoothed RTT to compute the latest smoothed RTT. We plan to investigate those issues related with dynamic membership changes in the future.

3.7 Summary

In this chapter, we elaborate on several mechanisms of the error recovery scheme: early packet loss detection, NAK suppression and aggregation, local recovery and scoped retransmission, hierarchical RTT measurement, and periodical ACK mechanism. We also discuss the details of congestion control scheme: congestion detection, congestion window regulation, and congestion smoothing buffer management. In addition, the potential problems and limitations of the error recovery scheme and the congestion control scheme of ARMC are also presented.

Chapter 4

Performance Evaluation

We use simulation to evaluate the performance of ARMC. We implement ARMC in NS2[NSV2] and use the PANAMA active package[PANA] to simulate the function of active networks. The whole evaluation is divided into four steps. The first is validation and verification of simulation. The second is a sensitivity test used to decide appropriate value of an important parameter. The other two parts are the performance evaluation of the error recovery scheme and the congestion control scheme respectively.

4.1 Validation and Verification of Simulation

It is important to validate and verify the correctness of the simulation. We validate the simulation using a 10 node topology. The extensive trace results are compared with the expected ones in several scenarios. The results show that the simulation works properly and correctly.

4.2 Sensitivity Test

The main purpose of a sensitivity test is to find appropriate value of some inherent parameters. The interesting parameter in this research is the time interval, Query Period (QP) for RTT, to send RTT-QUERY packets since the measured RTT affects both the error recovery scheme and the congestion control scheme. The objective is to find an appropriate value for the time interval to measure the newest RTT value so that the source and active routers

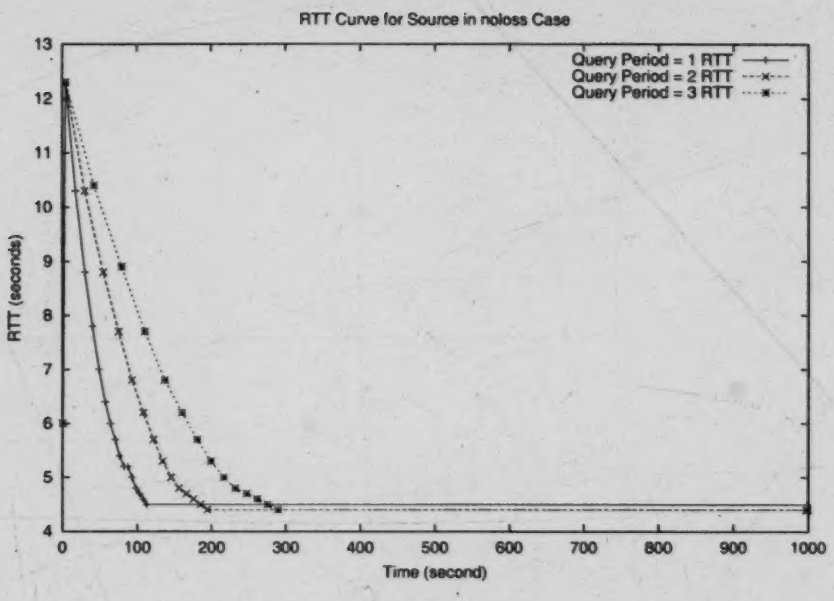
can keep track of the variation of RTT and also keep the number of RTT-QUERY packets and RTT-REPLY packets as small as possible.

The experiments are conducted on a 100 node transit-stub topology generated by GT-ITM[KM97]. The transit-stub topology is composed of transit domains which represent the backbone topology of the Internet and stub-domains which represent the edge network of the Internet. This type of topology closely matches the hierarchical nature of the Internet. The number of receivers is set to 15. Active routers are placed on the routers whose have 2 or more downstream links. We investigate how the RTT measurement mechanism behaves in two scenarios: NoLoss and Loss. The NoLoss case has no background flows. The Loss case is created by using 10 UDP (an unreliable unicast transport protocol on the Internet) flows as background traffic and randomly varying the source and destination of each UDP flow. In the Loss case, we do 5 sample runs, each of which may include different pairs of sources and destinations for each UDP flow.

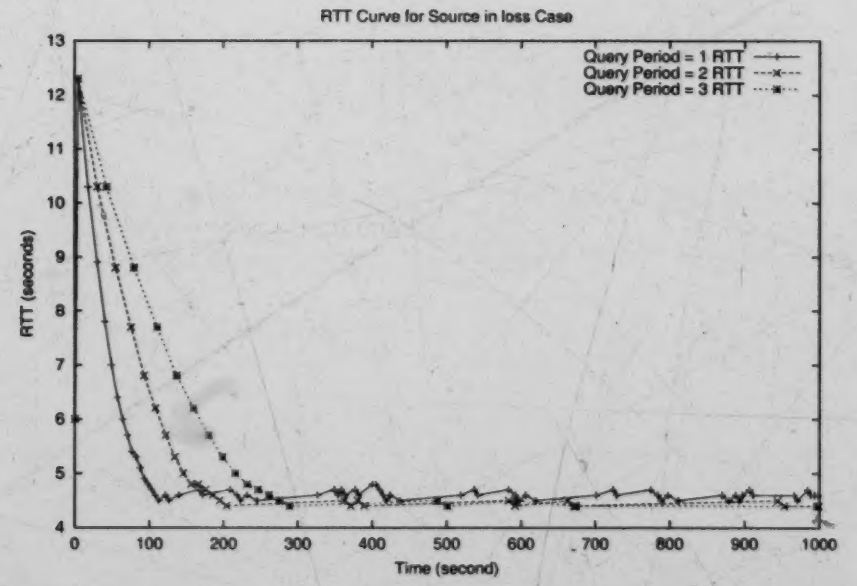
	No Loss	Loss
QP = 1 RTT	1065	1050
QP = 2 RTT	525	525
QP = 3 RTT	335	335

Table 4.1: Total Number of RTT_QUERY and RTT_REPLY Packets of Source for different query periods

Fig. 4.1 shows how the RTT measured in the source evolves with time in both the NoLoss case and one Loss case run for three different query periods. It indicates that the shorter the query period, the smaller the convergence time. Table 4.1 shows the total number of RTT_QUERY and RTT_REPLY packets in both the NoLoss case and one Loss case run for three different query periods. It indicates that the total number of RTT_QUERY and RTT_REPLY packets increases as the query period to measure the RTT become smaller. The other loss case runs are similar to the one presented. As a compromise, we choose 2 RTT as the time interval because it can converge to the real RTT quickly and only generate a medium number of RTT_QUERY and RTT_REPLY packets.



(a) No Loss



(b) Loss

Figure 4.1: RTT curve

4.3 Error Recovery

When we evaluate the error recovery scheme, we disable the congestion control scheme in order to avoid the interference and easily analyze the results. We first study the characteristics of our protocol. Then we compare ARMC with ARM[LS98] and AER/NCA[SS00]. The results show that our protocol can outperform both ARM and AER/NCA in almost all the tested cases.

4.3.1 Simulation configuration

We use the simple hierarchical RTT measurement mechanism to measure the upstream RTT and downstream RTT for active routers, receivers, and the source. The parameters used in the simulation are specified as follows.

- **Link bandwidth**: 1.5Mb
- **Link output queue size**: 50 (default value)
- **RTT query period**: 2 current measured largest downstream RTT
- **Timeout value for NAK records**: 1 upstream RTT
- **Timeout value for data packets and REPAIR records**: 1 largest downstream RTT

In all the experiments for the error recovery tests, we are mainly concerned with how retransmission buffer size and number of active routers can affect the performance of error recovery for different loss probability cases. Thus, as a first study, we use a single topology and the same multicast tree in all the experiments. The topology used in this test is a transit-stub topology with 200 nodes generated by GT-ITM[KM97]. We randomly choose 40 receivers and a source to construct a multicast tree. Receivers are only leaf nodes in the multicast tree. An active router and a receiver do not share the same node. We vary three factors: link loss probability, retransmission buffer size, and percentage of active routers to the total number of routers. Each link has the

same loss probability and the losses in each link are randomly distributed. To choose active routers, we favor the routers that have at least 2 downstream links because the number of receivers that they control is large and thus they potentially have more effect on the protocol performance. In each experiment, 40 packets are sent in one RTT and a total of 10000 packets are transmitted. For each experiment, we do 8 sample runs and average the results and compute 95% confidence interval.

The performance metrics of interest for the error recovery scheme are:

- **Average recovery time**: Recovery time is the time interval between the time when receivers are aware of a packet loss and the time when they receive the repair packet (the retransmitted missing packet). Average recovery time is the mean value of the recovery time of all the lost packets.

- **Nuisance**: Nuisance is defined as the ratio of the number of duplicate repair packets to total number of repair packets received by receivers. It is used to evaluate the effect of scoped retransmission.

We discuss the results from three perspectives. The first is the effect of buffer size on the performance. The second is the effect of number of active routers on the performance. The third is a comparison with ARM and AER/NCA. The vertical line in all the following graphs is the 95% confidence interval.

4.3.2 Effect of retransmission buffer size on performance

We first consider the results for the 35% active router case. In this case, all the routers with at least 2 downstream links are chosen as active routers. Because we vary the packet loss probability of every link in each experiment, the receivers may have different packet loss ratio due to their different distance from the source. Therefore, we would like to choose 3 typical receivers: farthest receiver, medium receiver, and nearest receiver, to present their results. We are also interested in the combined results of total receivers. Thus, for each

experiment, we present the results of 4 typical cases: farthest receiver, medium receiver, nearest receiver, and total receivers.

Fig.4.2 is the graph of Average Recovery Time vs. Loss Probability for 3 different retransmission buffer sizes, and Fig. 4.3 is the graph of Nuisance vs. Loss Probability for 3 different buffer sizes. The retransmission buffer stores both data packets and repair packets. From Fig. 4.2(d) and Fig. 4.3(d), we can see that the performance of the error recovery mechanism for all the receivers is proportional to the buffer size. Basically, as the retransmission buffer size increases, the average recovery time and nuisance will decrease. Obviously, the reason is that those packets that can not be buffered due to limited buffer space would be buffered in the large buffer case. Then the recovery time would be improved if those packets are lost in the downstream links. Increased buffer space can also make it possible for the repair packets that are not buffered in the small buffer size case to be stored which reduces the number of subcasts since the repair packet will be subcast if there is no buffer space when it arrives at the active router. Increased buffer size finally reduces the possibility that receivers receive duplicate repair packets and thus decreases the nuisance. This shows the effectiveness of local recovery. Although it can enhance the performance by increasing the retransmission buffer size, the performance improvement is limited when buffer size is increased beyond some value. The reason is that at some buffer size almost every packet can be buffered. At that time, most of the increased buffer space is unused if buffer size is further increased. The number of buffered packets nearly doubles when the buffer size is increased from 20 to 40. This is because the total number of packets sent in one RTT consumes 40 buffers in the active router. However, when the buffer size is increased from 40 to 80, the repair packets occupy most of the increased buffer space. The number of repair packet is small, so the performance improvement also is limited. As the loss probability increases, the average recovery time and nuisance also increase slightly. Although the buffer size remains the same as the loss probability increases, the number of repair packets increases and most of them can not be buffered due to limited

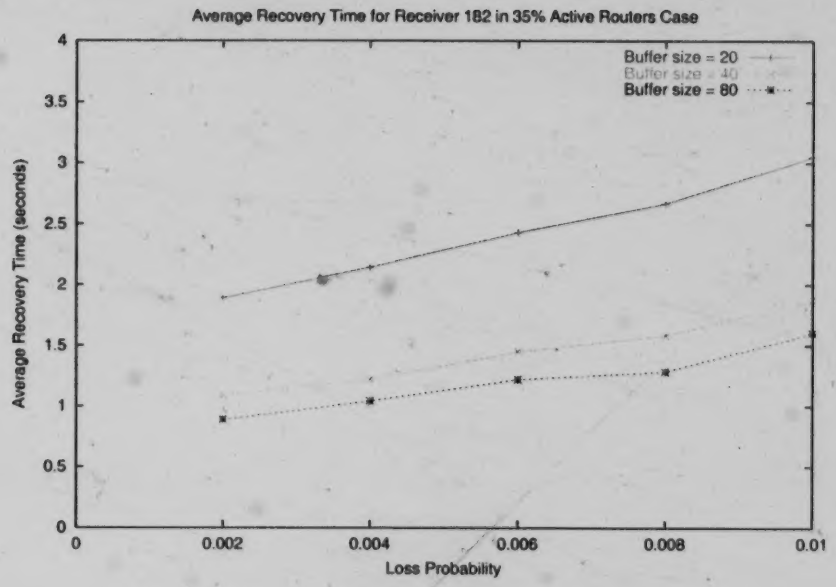
buffer space increases. This can explain why the performance is slightly worse as the loss probability becomes greater.

We also investigate the performance of particular receivers. Since we vary the packet loss probability of every link in each experiment, the receivers may have different packet loss probability due to the different distance from the source. For example, in the case that the link loss probability is p, the perceived loss probability by a receiver whose distance to the source is n is $1-(1-p)^n$. Then the longer the distance between a receiver and the source, the greater the perceived loss probability. From Fig. 4.2 and Fig. 4.3, we can see that the performance for farthest receiver and medium receiver is similar to that of total receivers. However, it seems wrong that the nearest receiver sees no performance change when the buffer size varies. Nevertheless, this is reasonable because there are no active routers between the source and the nearest receiver so that the change of buffer size has no effect on its performance. In addition, we also vary the percentage of active routers to investigate the performance when we change the buffer size in active routers. However, the results for different percentage of active routers are qualitatively similar to those in the 35% active routers case. Therefore, we do not present the results for those experiments but will investigate this effect in the next experiment for the buffer size of 40.

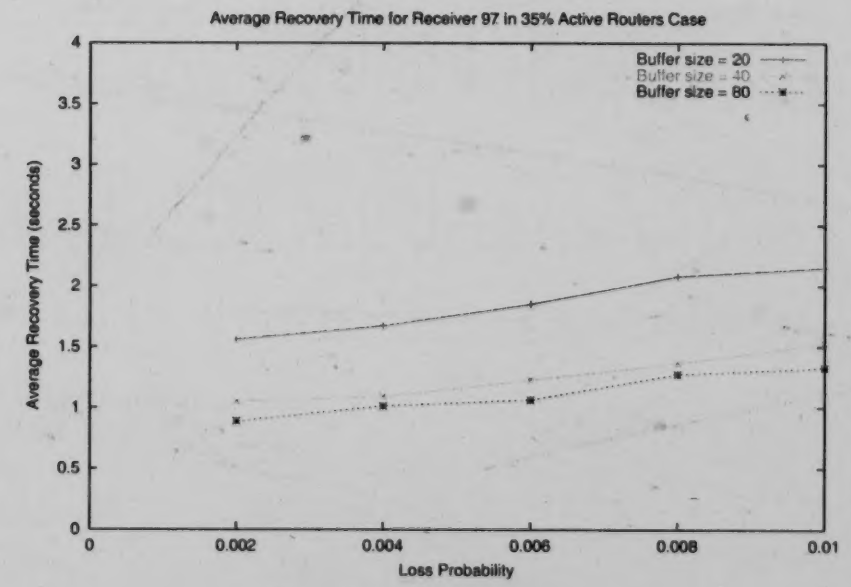
4.3.3 Effect of number of active routers on performance

We vary the number of active routers by varying the percentage of the number of active routers to the number of total routers in the multicast tree. We first consider the case when retransmission buffer size is 40 since the results in the last section show that good performance can be achieved and only medium buffer space is used in this case. Fig. 4.4 shows Average Recovery Time vs. Loss Probability for different numbers of active routers. Fig. 4.5 shows Nuisance vs. Loss Probability for different number of active routers.

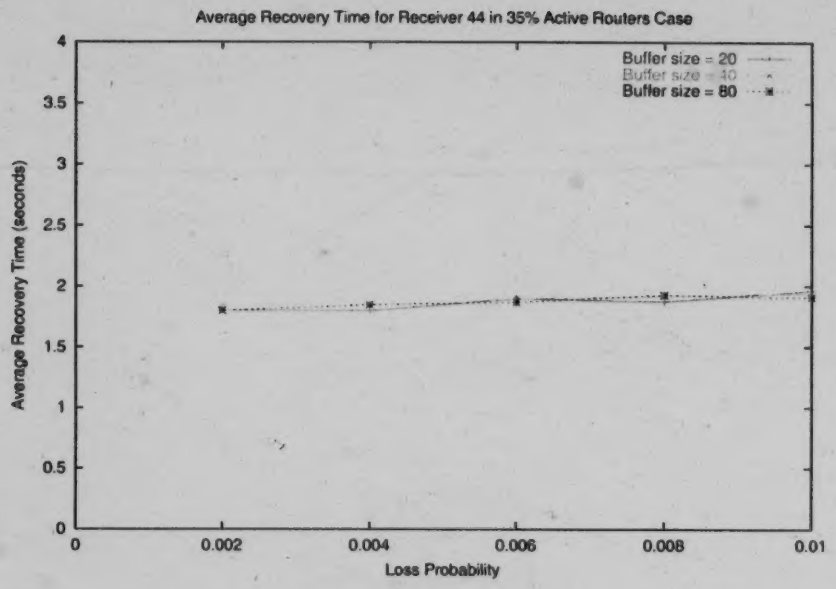
Fig. 4.4(d) and 4.5(d) show the results averaged over all the receivers. Fig. 4.4(d) indicates that the average recovery time is slightly improved as the



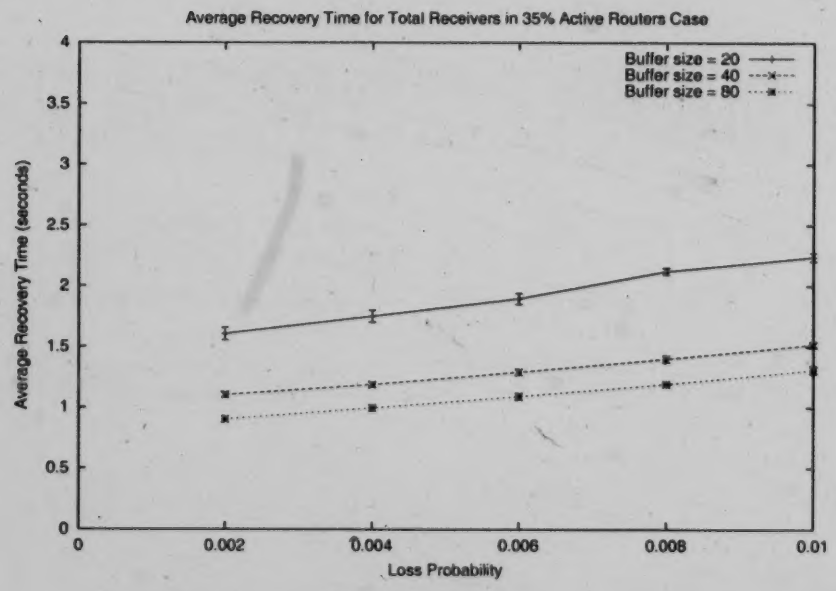
(a) farthest receiver



(b) medium receiver

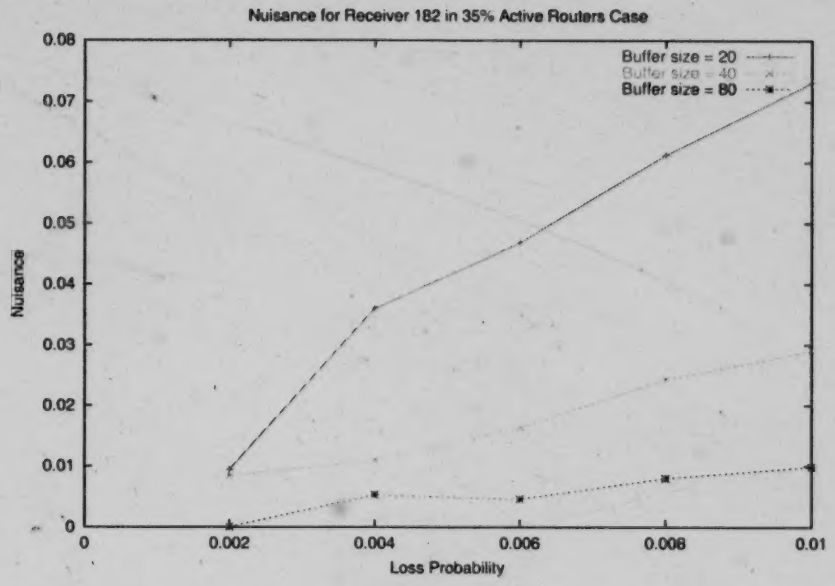


(c) nearest receiver

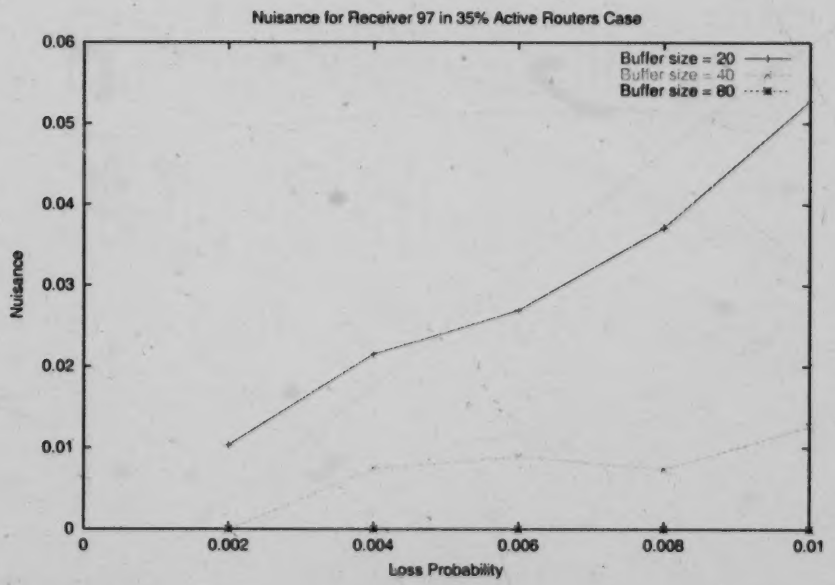


(d) total receivers

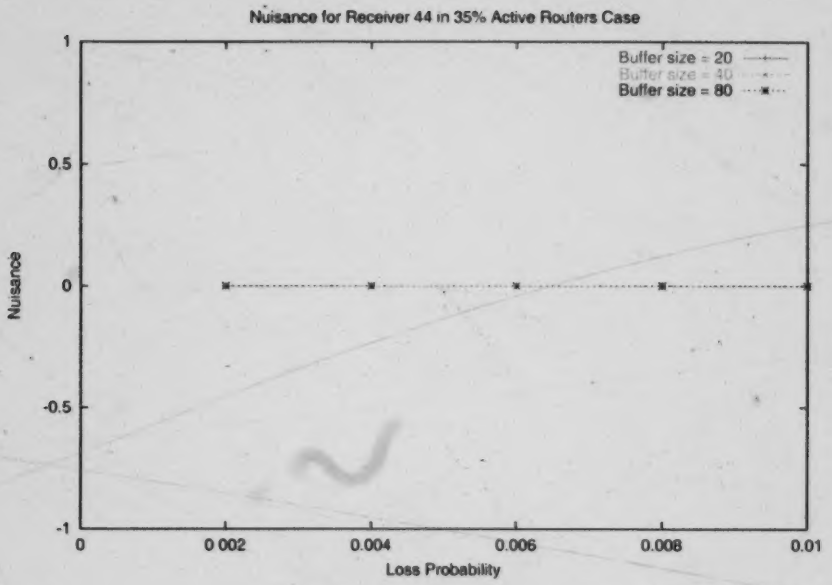
Figure 4.2: Average recovery time in 35% active routers case. All the routers that have at least 2 downstream links are chosen as active routers.



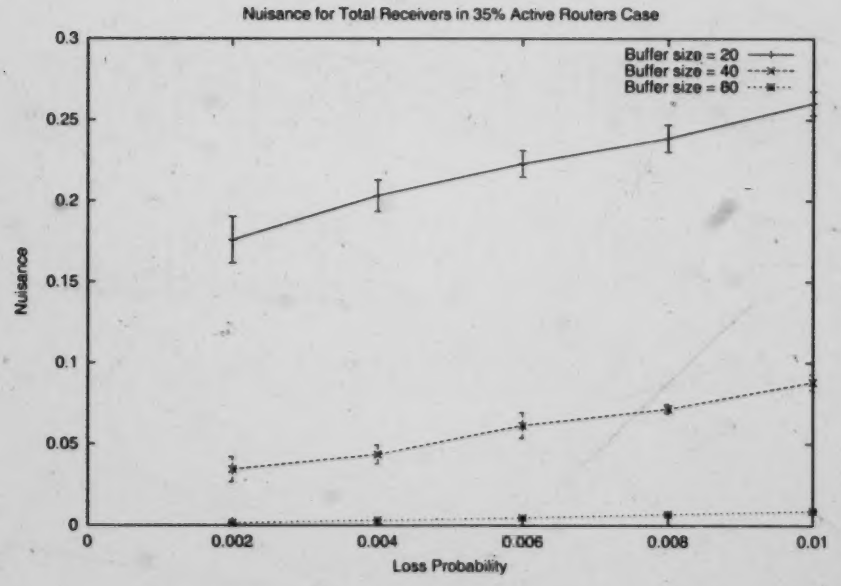
(a) farthest receiver



(b) medium receiver

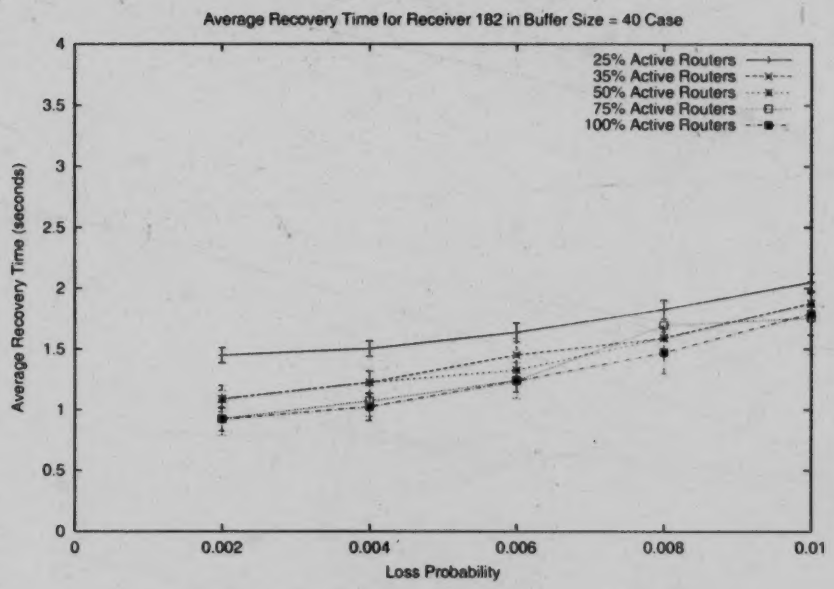


(c) nearest receiver

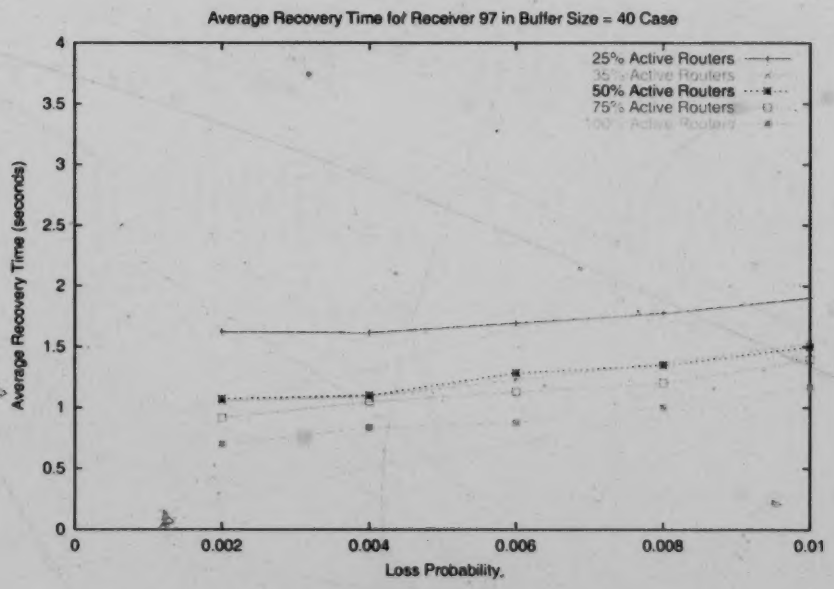


(d) total receivers

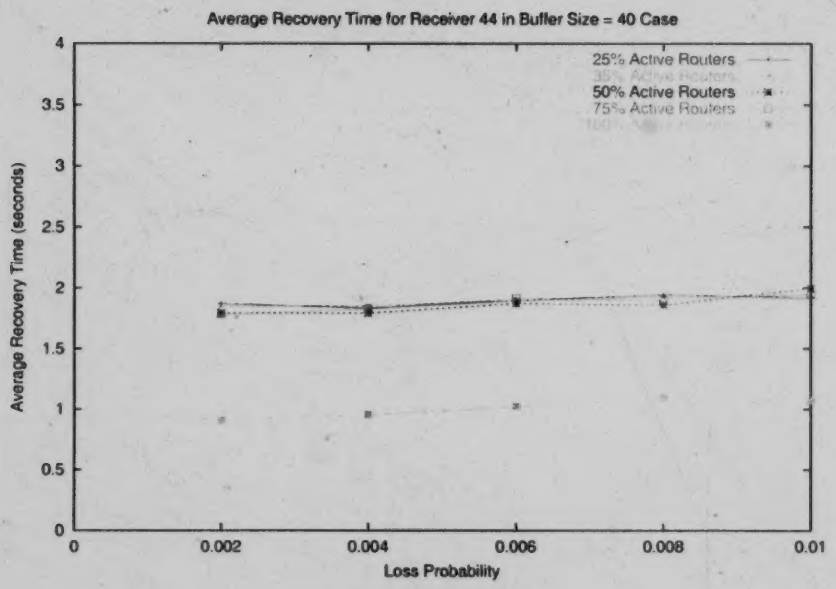
Figure 4.3: Nuisance in 35% active routers case. All the routers that have at least 2 downstream links are chosen as active routers.



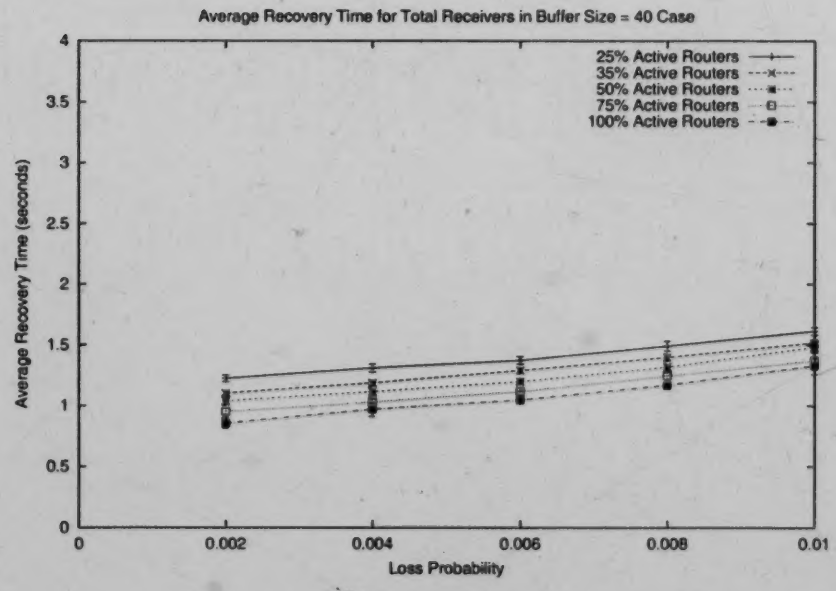
(a) farthest receiver



(b) medium receiver

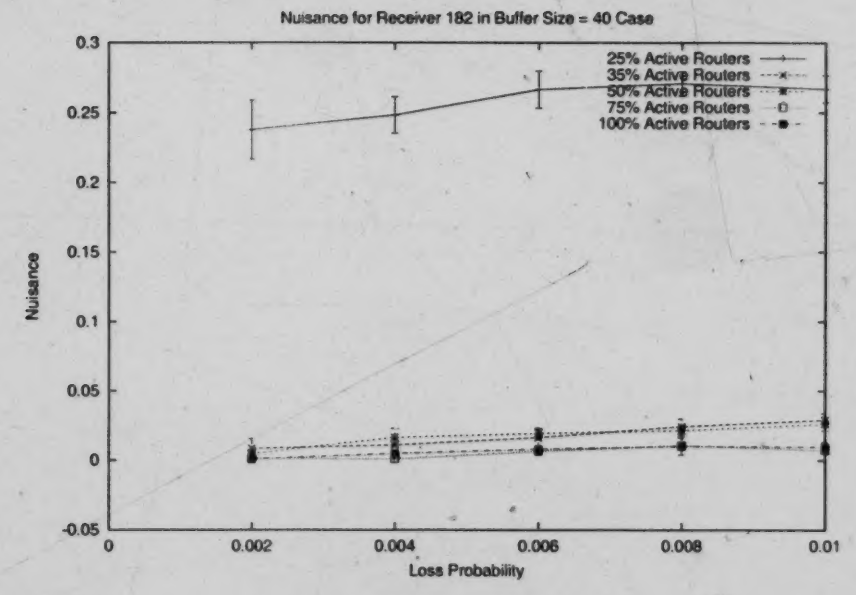


(c) nearest receiver

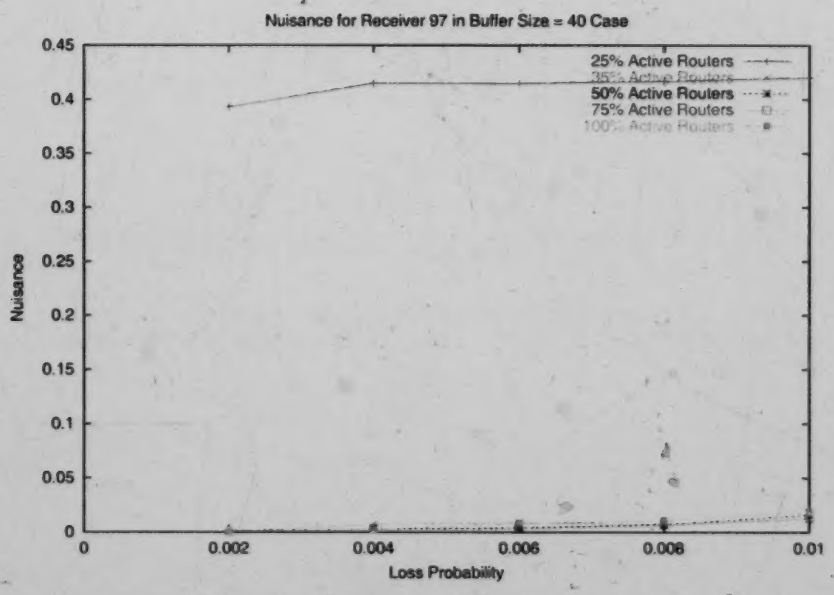


(d) total receivers

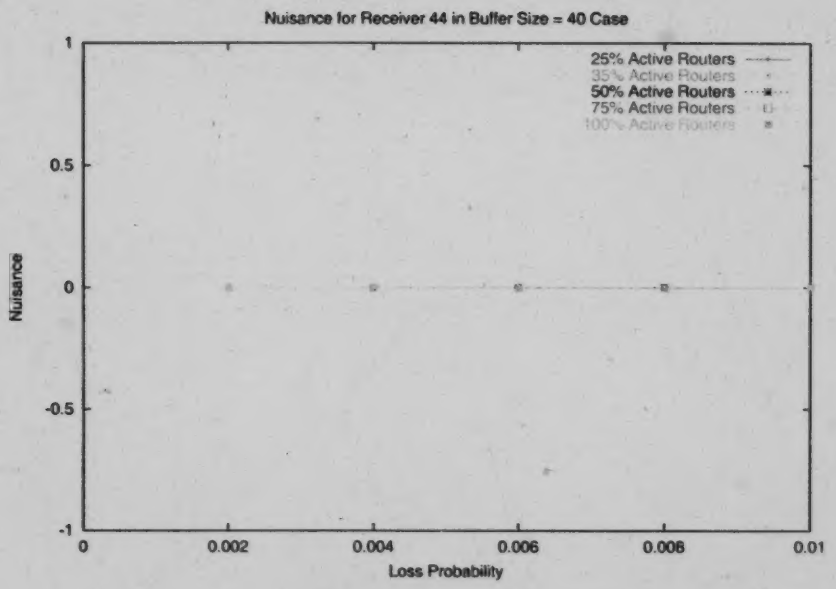
Figure 4.4: Average recovery time for total receivers in buffer size = 40.



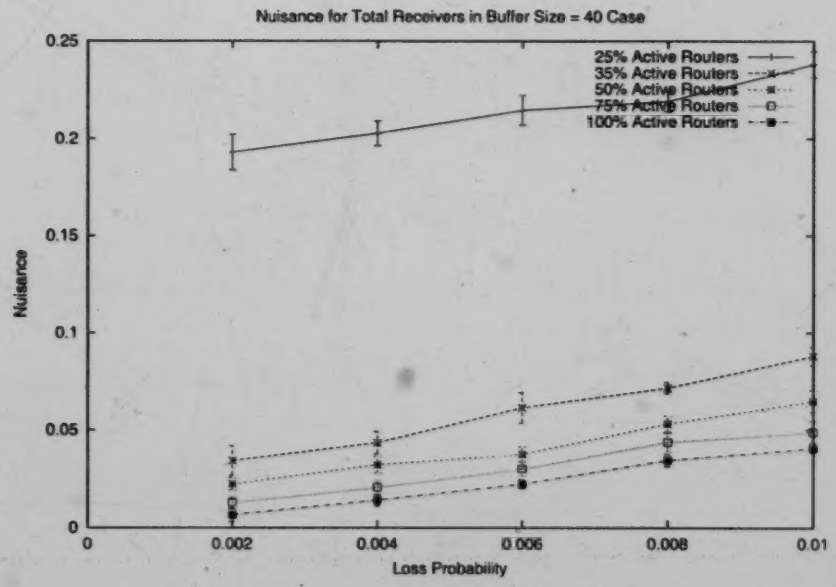
(a) farthest receiver



(b) medium receiver



(c) nearest receiver



(d) total receivers

Figure 4.5: Nuisance for total receivers in buffer size = 40

number of active routers increases. Fig. 4.5(d) indicates that the nuisance is significantly enhanced between the case of few active routers (25%) and the case where each active router has 2 or more downstream links (35%), while the nuisance is only modestly improved when the number of active routers continuously increases. Therefore, considerable performance enhancement can be achieved when active routers are placed on some strategic points (in this test, at the 35% router case, the routers that have at least 2 downstream links are chosen as active routers) while the total number of active routers remains a small size. Fig. 4.4(a) and 4.5(a), Fig. 4.4(b) and 4.5(b), and, Fig. 4.4(c) and 4.5(c) show the Average Recovery Time and Nuisance respectively for the farthest receiver, the medium receiver, and the nearest receiver. They show that increasing the number of active routers may not necessarily improve the performance of a specific receiver. The reason is that the number of active routers in the path from a specific receiver to the source may not increase when the total number of active routers increases. Similar to the former section, we also present only the above representative graphs because other experiments are qualitatively similar.

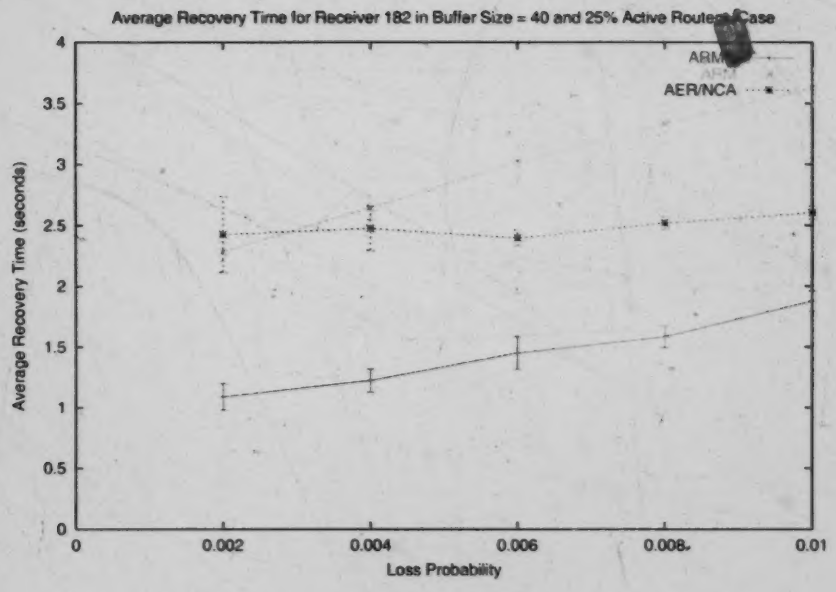
4.3.4 Comparison of ARMC , ARM, and AER/NCA

In this section, we compared ARMC, ARM and AER/NCA. We directly use the simulation software of AER/NCA provided by the authors at the University of Massachusetts, Amherst. All the parameters in AER/NCA software remains unchanged. In this experiment, we disable their congestion control scheme in order to compare with just the error recovery scheme. We implemented ARMC and ARM by ourselves. We set the value of parameters in ARM to recommended ones in [LS98]. In all experiments, we set the retransmission buffer size to 40 and only vary the number of active routers in ARMC, ARM, and AER/NCA. We do not control the buffer size of NAK records or REPAIR records in ARMC and ARM or the buffer size of pending NAK records in AER/NCA. We first consider the combined results for total receivers in the 35% active routers case since the previous results indicate that considerable

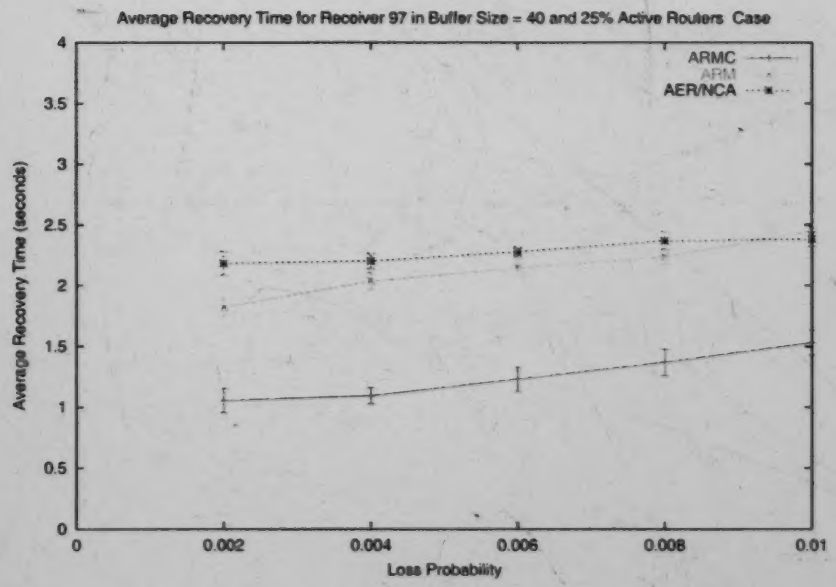
performance enhancement can be achieved with this small number of active routers. In this test, we add one more metric, **number of NAKs sent by receivers**, to evaluate the effect of our NAK suppression mechanism.

Fig. 4.6 shows Average Recovery Time vs. Loss Probability and Fig. 4.7 shows Nuisance vs. Loss Probability. Fig. 4.6(d) and 4.7(d) are the combined results for all the receivers. They indicate that ARMC outperforms ARM and AER/NCA in both average recovery time and nuisance. AER/NCA uses a randomized timer mechanism to suppress NAKs with the same packet sequence number sent by active routers or receivers. This suppressing mechanism makes active routers or receivers wait a random amount of time before sending NAKs and initiating a recovery process even if they are the first node that detects the packet loss. In ARMC, active routers or receivers send NAKs to upstream nodes immediately once they detect packet loss. At the same time, active routers also send FAKE_DATA packets to suppress NAKs that would be potentially sent by downstream nodes. That is possibly why ARMC attains a better recovery time than AER/NCA. In ARM, only receivers can check sequence number gaps and send NAKs. This mechanism makes ARM take a longer time to find packet loss and initiate a recovery process. This is why ARM has a greater average recovery time than ARMC.

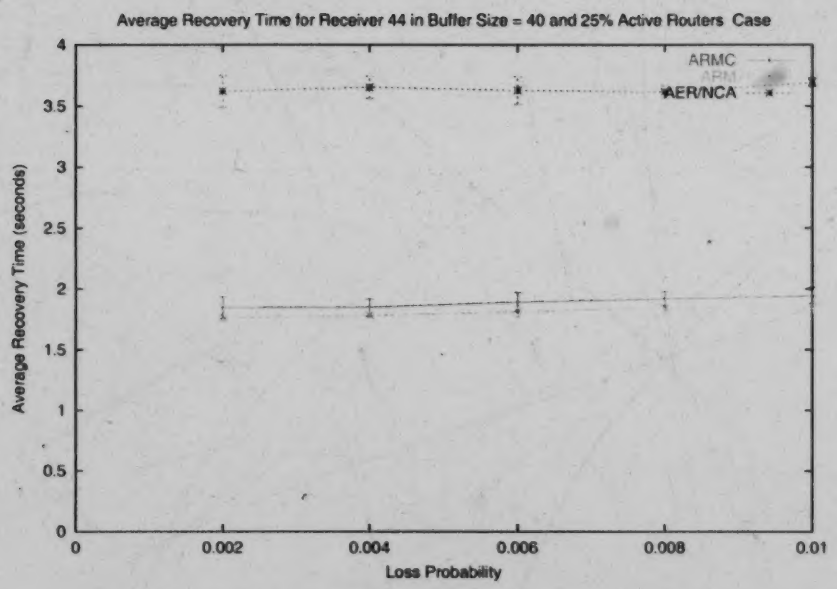
ARMC also has a better nuisance value than AER/NCA. In AER/NCA, each time when an active router or the source sends a repair packet, they subcast it to the whole subtree. In ARMC, each active router keeps a NAK subscription bitmap in the NAK record. When it sends repair packets, it sends them only to the downstream links from which NAKs come. This accounts for why ARMC has a significantly lower nuisance value than AER/NCA. The drawback is that the soft-state storage to buffer the NAK records needs to consume some memory. However, the size of each NAK record is quite small. In addition, this results in significant bandwidth reduction consumed by duplicate repair packets. Though ARM also uses a similar NAK subscription bitmap to limit the scope of repair packets, it has no way to learn the RTT value for the timer. In these experiments, we set the timeout value in active



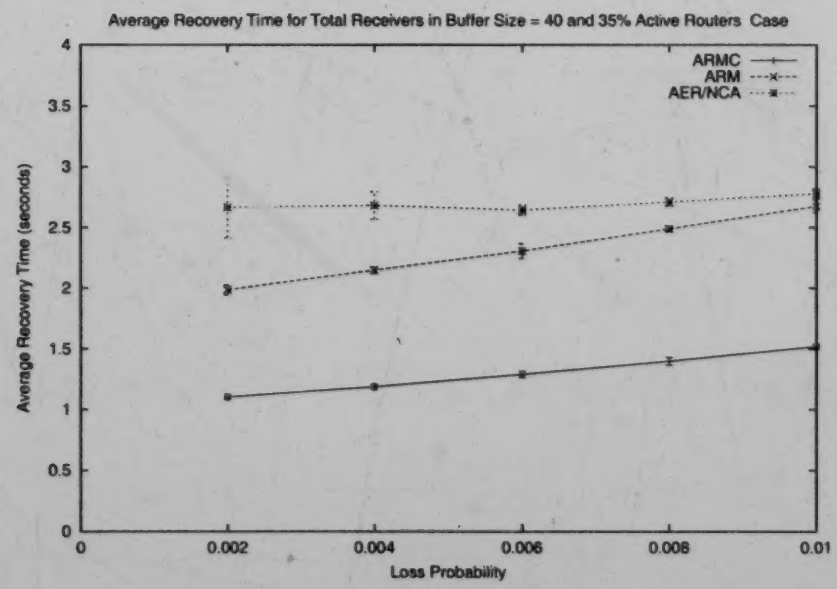
(a) farthest receiver



(b) medium receiver

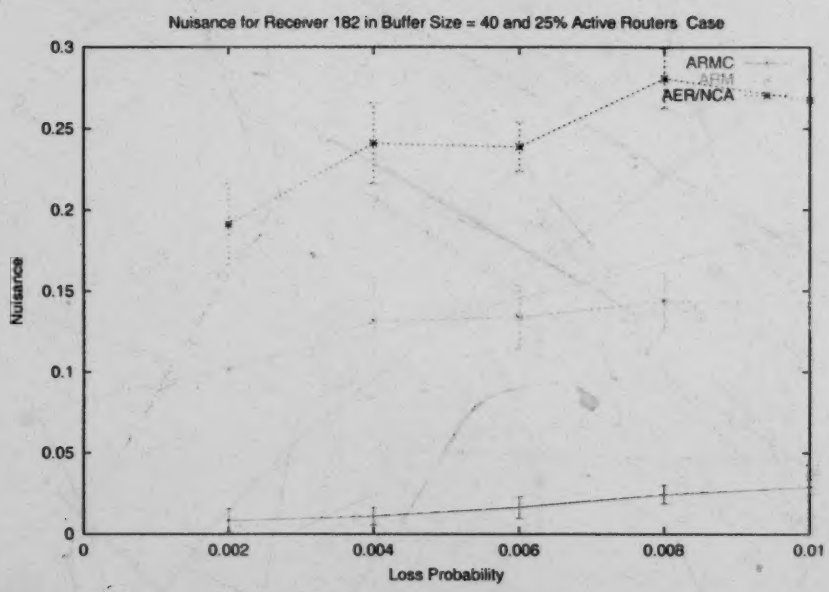


(c) nearest receiver

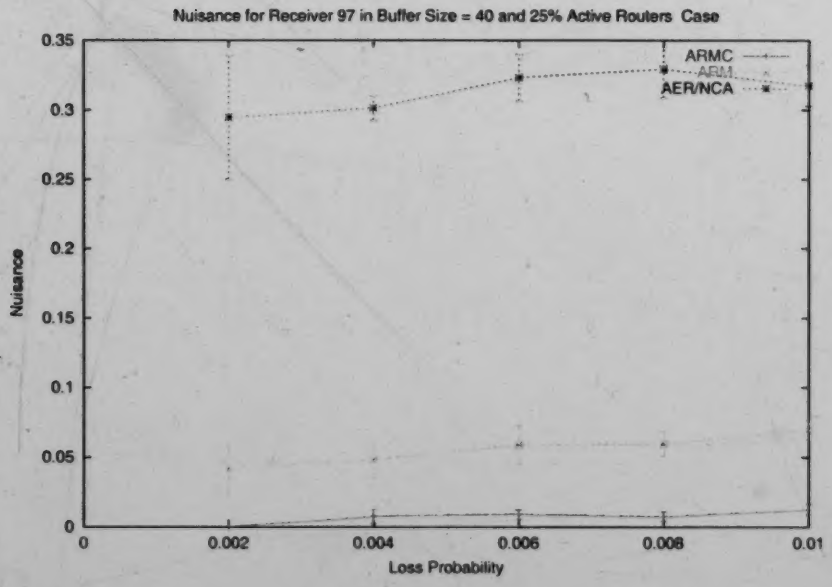


(d) total receivers

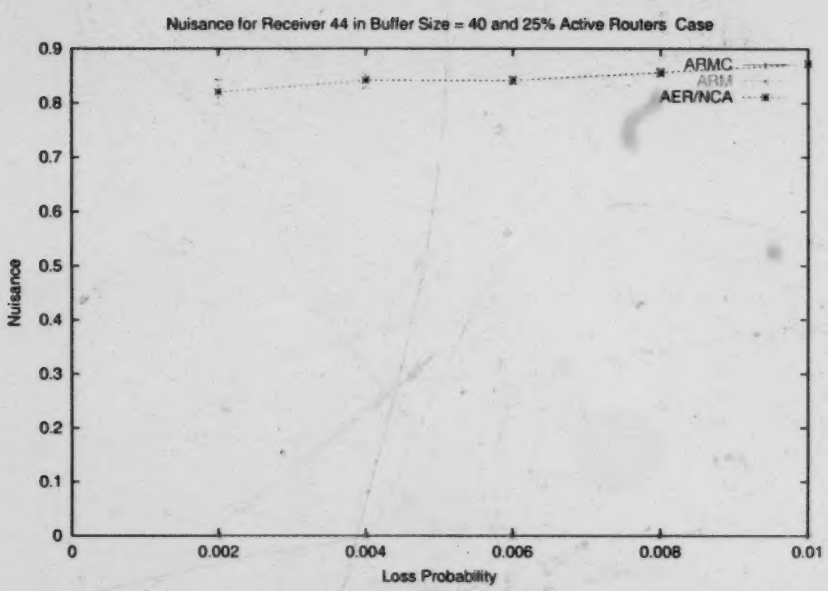
Figure 4.6: Comparison of ARMC, ARM and AER/NCA. Average recovery time in 35% active routers and buffer size = 40.



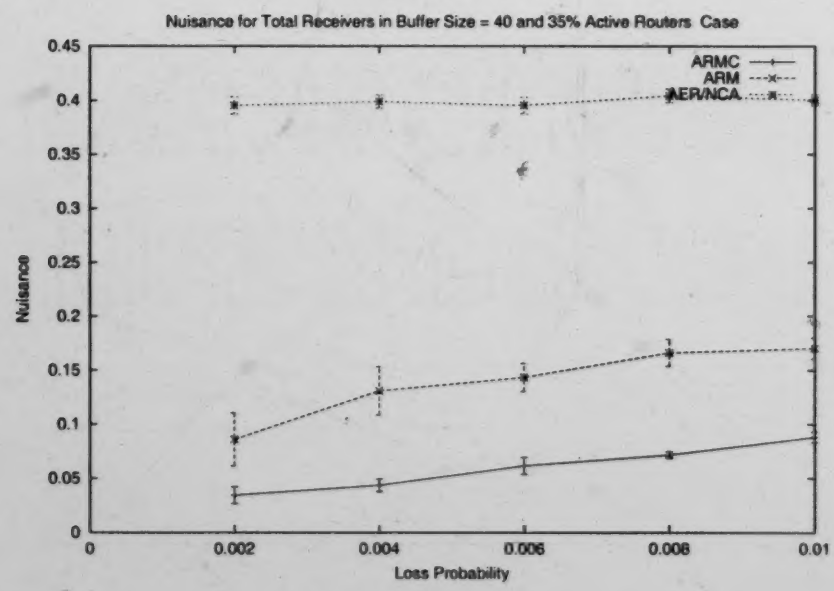
(a) farthest receiver



(b) medium receiver



(c) nearest receiver



(d) total receivers

Figure 4.7: Comparison of ARMC, ARM and AER/NCA. Nuisance in 35% active routers and buffer size = 40.

routers to be the largest RTT from the source to the farthest receiver as recommended in ARM[LS98]. Thus, the buffered time of data packets is a bit longer than that in ARMC. This may increase the number of subcasting repair packets due to unavailable buffer space when they arrive at active routers. That is possibly the reason why nuisance in ARMC is a little better than in ARM. From Fig. 4.6 and Fig. 4.7, the results for the farthest receiver and medium receiver are similar to those of all the receivers. The results for the nearest receiver are a little different from others between ARMC and ARM. The reason is that there are no active routers between the source and the nearest receiver. In addition, the estimated RTT from the source to the nearest receiver is almost the same in both ARMC and ARM. Then the error recovery time for the nearest receiver in ARMC and ARM is similar. The ARM also uses a subscription bitmap to limit the retransmission scope. Furthermore, the nearest receiver directly connects with the source and each packet loss is recovered immediately by the source. Therefore, the nuisance in ARM is almost the same as that in ARMC. Similar to the former section, we do not present the graphs for different percentages of active routers due to the qualitative similarity.

Loss Prob.	0.002	0.004	0.006	0.008	0.01
ARMC	1365	2851	4468	6198	8070
ARM	3774	7257	11042	15308	19503
AER/NCA	1607	5098	8280	12361	18450

Table 4.2: Number of NAKs Sent by All the Receivers

Loss Prob.	0.002	0.004	0.006	0.008	0.01
ARMC	2166	4551	6773	8957	10977
ARM	0	0	0	0	0
AER/NCA	5995	18048	28603	48936	67555

Table 4.3: Number of Overhead Packets for NAK Suppression. In ARMC, it is the number of FAKE_DATA packets received by all the receivers. In AER/NCA, it is the number of NAKs received by all the receivers

Table 4.2 shows the number of NAKs sent by all the receivers for the 35% active router case under different loss probability conditions. it indicates that

ARMC has a smaller number of NAKs sent than ARM and AER/NCA. Table 4.3 shows the number of overhead packets for NAK suppression. ARMC uses FAKE_DATA to suppress NAKs. ARM has no NAK suppression mechanism. This is why the number of NAKs sent by ARMC is better than ARM and the number of overhead packets in ARMC is worse than ARM. In ARMC, the decrease in the number of NAKs sent by receivers is at the cost of the increase of the number of FAKE_DATA packets received by receivers. The sum of the number of NAK packets and overhead packets is similar for ARMC and ARM. In other words, we reduce the processing cost in active routers and the bandwidth consumption on the reverse paths from receivers to the source by increasing the bandwidth consumption on the paths from the source to the receivers. Also note that ARMC has lower average recovery time, which makes it more attractive than ARM.

The number of NAKs sent in ARMC is also less than that in AER/NCA. Although AER/NCA has a random timer-based suppression mechanism, for the packets lost in intermediate links, it does not necessarily suppress every NAK sent by receivers due to its random nature. For the packets lost in end links, suppression is useless if the lost packets are isolated, that is packets lost on different links are different. For example, packet $p1$ is lost on the link from node $n1$ to node $n2$ while packet $p2$ is lost on the link from node $n1$ to node $n3$ though the number of packets transmitted on them is the same. In this case, the suppression packet for $p2$ sent by node $n1$ to node $n2$ is useless since packet $p2$ does not miss on the link from node $n1$ to node $n2$. In addition, in the case of isolated loss in different end links, a timer-based suppression mechanism has another disadvantage in that the number of NAKs received by receivers is significantly increased since NAKs lost in each receiver have to be subcast to other receivers to suppress non-existent NAKs. In the experiment, the packet loss on the link is randomly distributed. Therefore, it is possible that most of the packets lost in different links are different. In AER/NCA, each time when an active router receives a NAK, it will subcast it to the downstream nodes if no repair packet is available in active routers.

This may account for why AER/NCA has a larger number of NAKs sent and a significantly larger number of overhead packets than ARMC. Moreover, the timer-based suppression mechanism in active routers and receivers also increases the processing cost.

4.3.5 Summary for error recovery

In this section, we discuss the performance evaluation of the error recovery scheme of ARMC. The results are presented from three perspectives: effect of retransmission buffer size on the performance, effect of number of active routers on the buffer size, and performance comparison of ARMC, ARM, and AER/NCA. They show that the the error recovery scheme can achieve considerable performance improvement with limited number of active routers and outperforms ARM and AER/NCA in almost all the cases tested.

4.4 Congestion Control

For studying the performance of the congestion control scheme, we are mainly concerned with how buffer size can affect the bandwidth sharing among different multicast and unicast flows. We are interested in how the throughput (receiving rate) or average throughput (average receiving rate) of a node evolves with time. Throughput is averaged over 5-second intervals at a node. Average throughput is the average of throughput computed over the interval from the simulation start point to the time of calculation. The results are presented from three perspectives. The first is the interaction between multicast flows and TCP flows. The second is the interaction among multiple multicast flows. The third is the study of inter-receiver fairness.

4.4.1 Interaction with TCP flows

Fig. 4.8 is the topology used in this experiment. Each link has 1 Mb bandwidth and uses a Droptail queue whose size is 50. The dashed lines in the graph construct a multicast tree. There are two active routers in the multicast tree

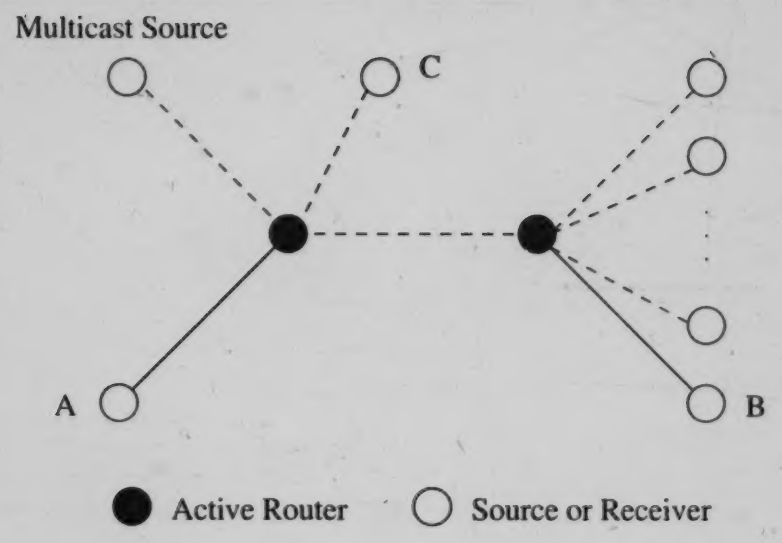
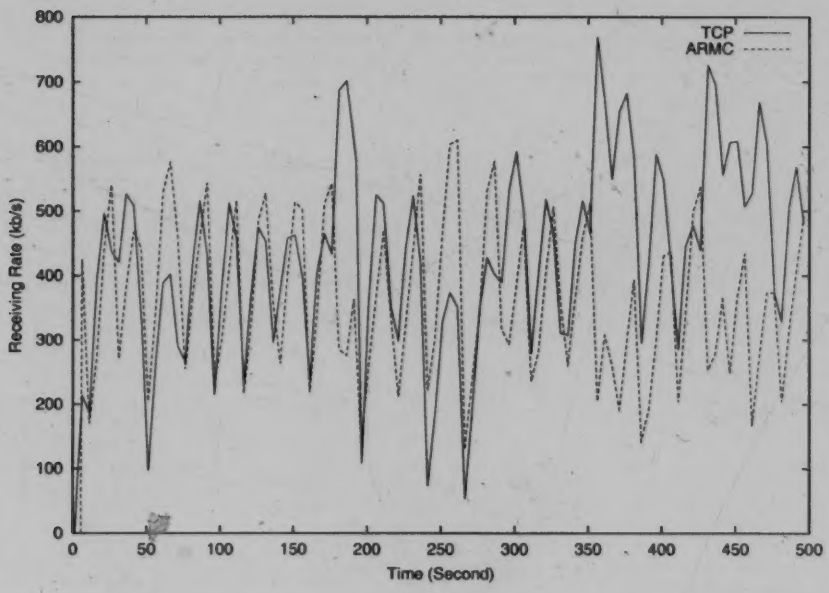


Figure 4.8: Topology

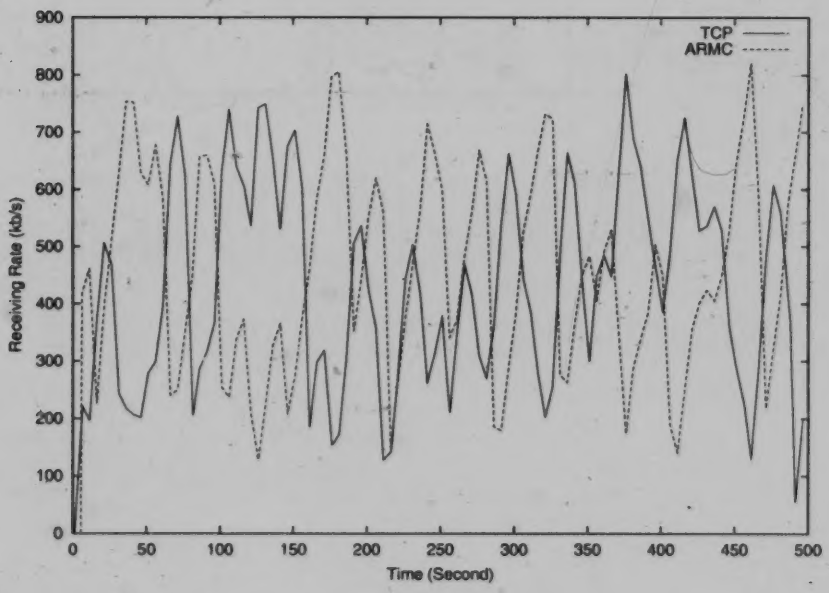
as illustrated in the graph. We run two experiments on this topology by changing the destination of the TCP flows. In these experiments, we vary the buffer size to see how a multicast flow competes for bandwidth of the congested link with a TCP flow. The multicast flow and TCP flow are both modeled as having infinite data to send and never terminate. The experiments last 500 seconds.

In the first experiment, a TCP flow is set up along the longest path of the multicast tree between node A and node B. The RTT of the TCP flow and that of the multicast source are both around 230ms. The buffer size in active routers is controlled by varying the α, the estimated congestion duration in RTTs of a nearest upstream active router. Fig. 4.9 shows the Receiving Rate vs. Time for different α values. Fig. 4.10 shows the Average Receiving Rate vs. Time for different α values.

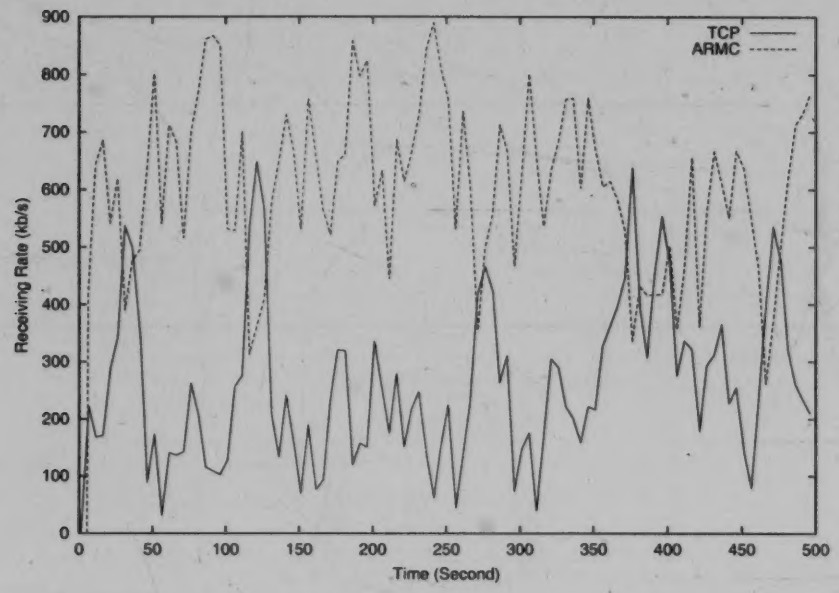
Fig. 4.9 and Fig. 4.10 indicate that the multicast flow can almost fairly share the congested link bandwidth with the TCP flow when the congestion buffer size is 0 ($\alpha = 0$). In this case, the active router is only responsible for



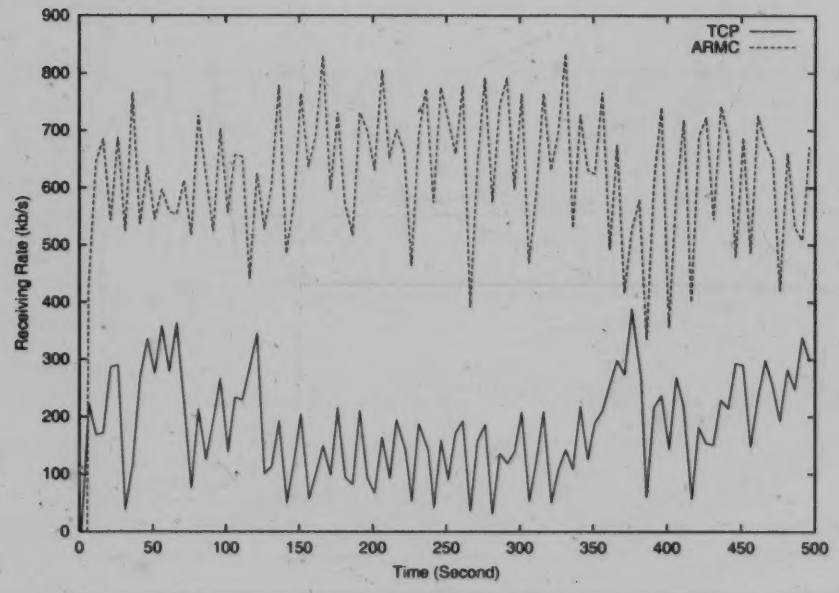
(a) $\alpha = 0$



(b) $\alpha = 2$

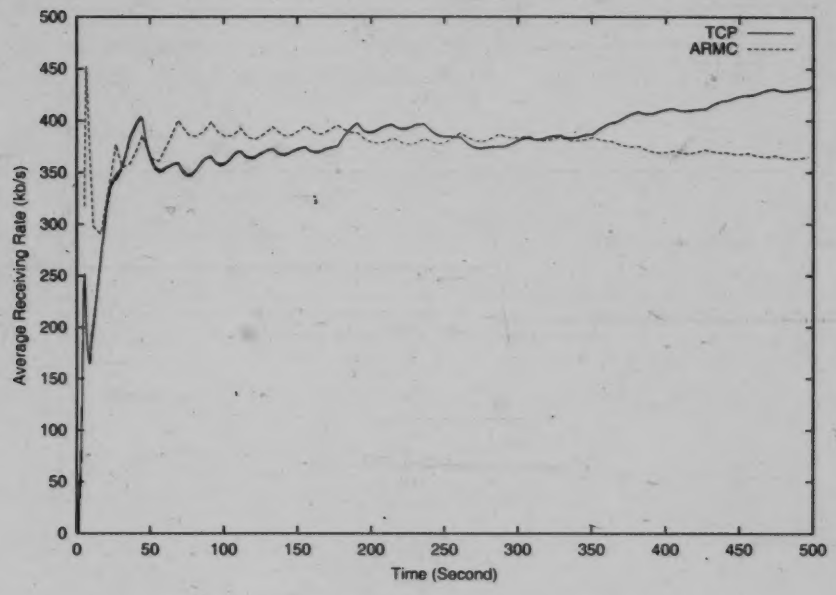


(c) $\alpha = 10$

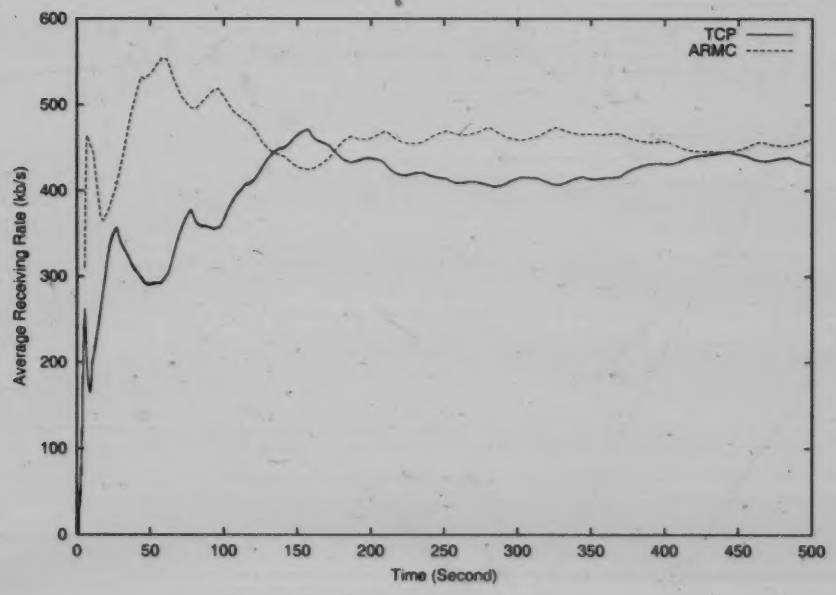


(d) $\alpha = 100$

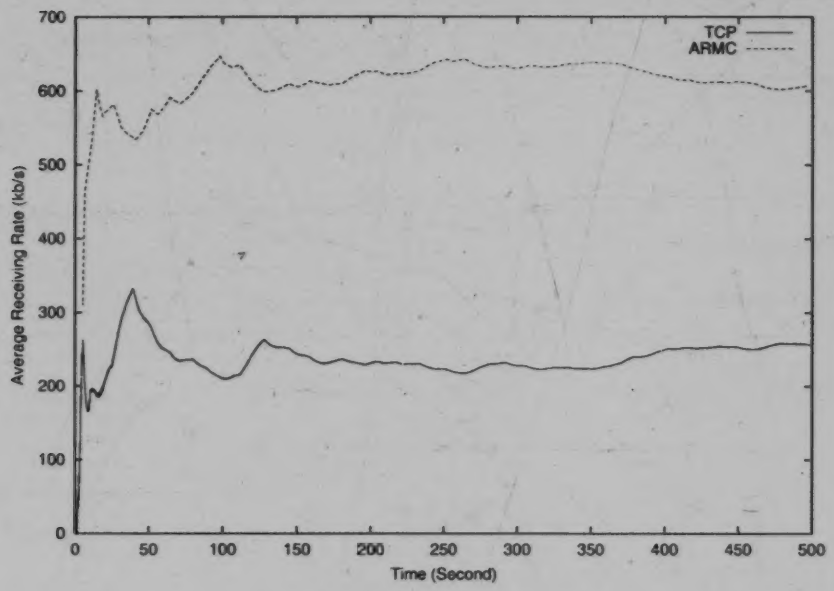
Figure 4.9: Receiving rate when a multicast flow and a TCP flow interact along the longest path. A multicast source has the same RTT as the TCP flow.



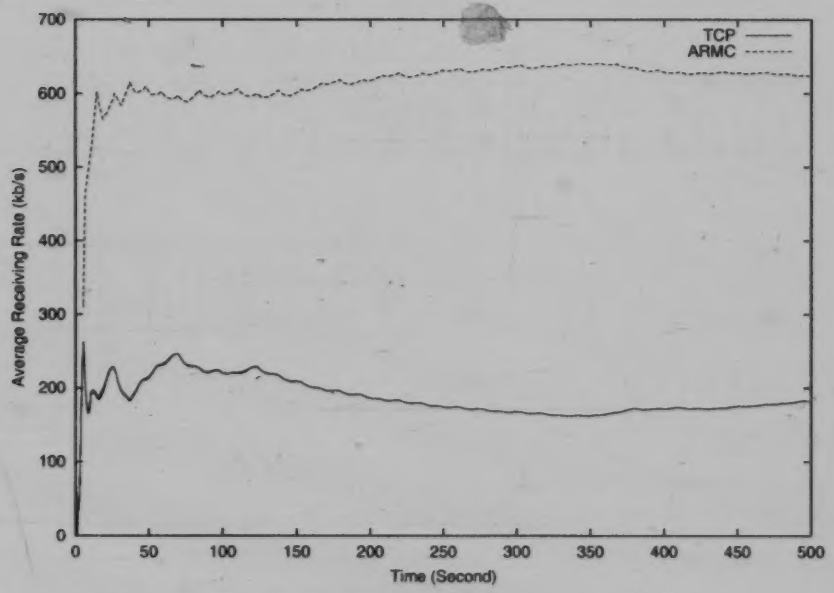
(a) $\alpha = 0$



(b) $\alpha = 2$



(c) $\alpha = 10$

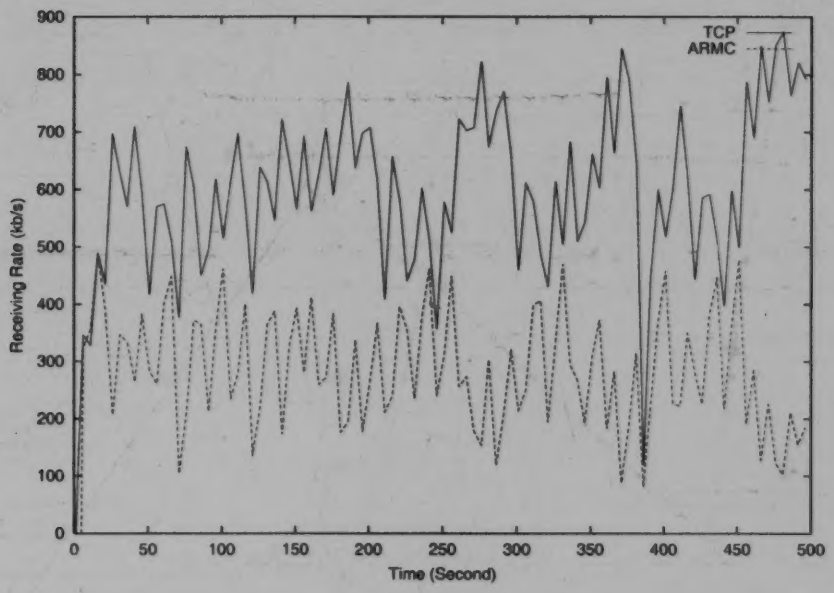


(d) $\alpha = 100$

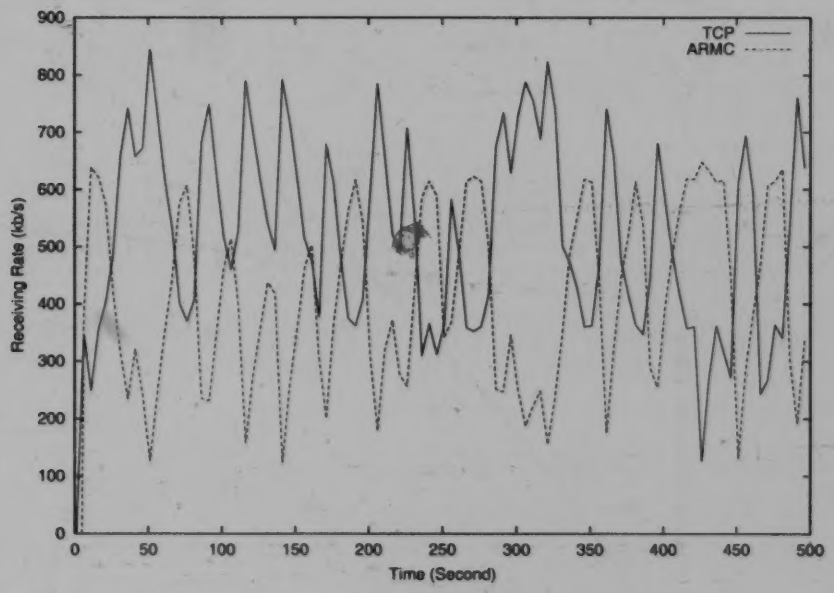
Figure 4.10: Average receiving rate when a multicast flow and a TCP flow interact along the longest path. A multicast source has the same RTT as the TCP flow.

aggregating the NAKs or CISes and guarantees that at most one NAK or CIS is forwarded to upstream nodes in one RTT. Since the multicast source reacts to packet loss at most once in one RTT and uses a similar window regulation algorithm as TCP, they can compete fairly when they have a similar RTT. Although the timeout reaction in TCP can sometimes prevent TCP from achieving high throughput, TCP has a strong mechanism to be immune to the rate reduction caused by out-of-order packets. In addition, TCP may take a longer time to detect the congestion than ARMC since it uses an ACK-based and timeout-based mechanism. Therefore, though our window regulation mechanism does not have a timeout mechanism, our mechanism does not grab more bandwidth than TCP. However, as the congestion smoothing buffer size in active routers increases, the multicast flow will grab more bandwidth than TCP. This is understandable because the active router now controls the rate regulation mechanism when it has buffers. The intermediate active router has a smaller RTT than the competing TCP connection. So it tends to increase the window size faster than TCP though at the same time it may also decrease its rate faster than TCP. In this case, it is much like the competition case between a TCP connection with a short RTT and a TCP connection with a long RTT. It is also worth noting that the ability for a multicast flow to grab bandwidth is limited. When the buffer size is up to some value, ARMC is much like a relay mechanism where the source sends data to the active router which is totally responsible for regulating the rate and forwarding the data. Since the active router also uses a similar window regulation mechanism with TCP, the rate difference is mainly due to the difference between RTT which is limited. This guarantees that a multicast flow never starves a TCP flow.

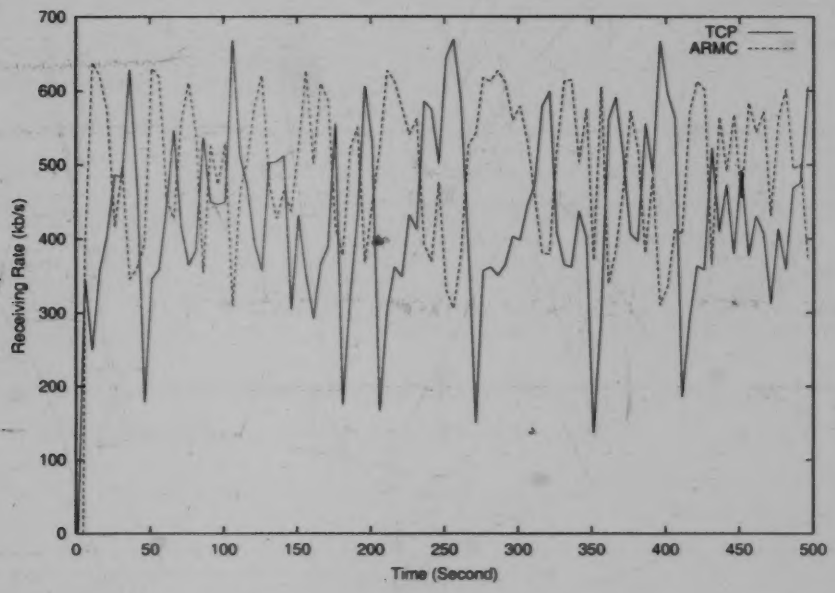
In the second experiment, a TCP flow is running between node A and node C. The latency for the TCP flow is around 160ms; while the RTT for the multicast source remains around 230ms. Fig. 4.11 and Fig. 4.12 are respectively Receiving Rate vs. Time and Average Receiving Rate vs. Time for this case. They show that the TCP flow can grab more bandwidth than the multicast flow in the case where α is 0. The reason is that the TCP flow has a shorter



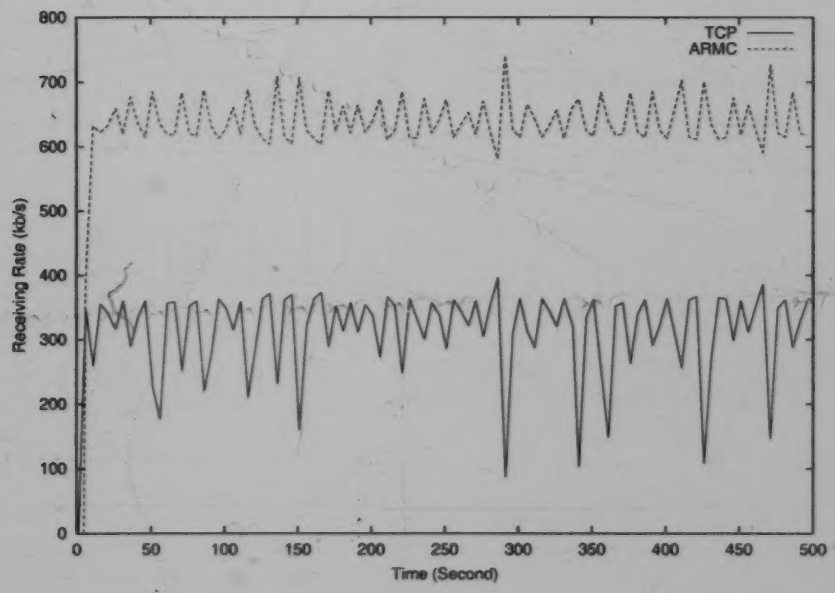
(a) $\alpha = 0$



(b) $\alpha = 2$

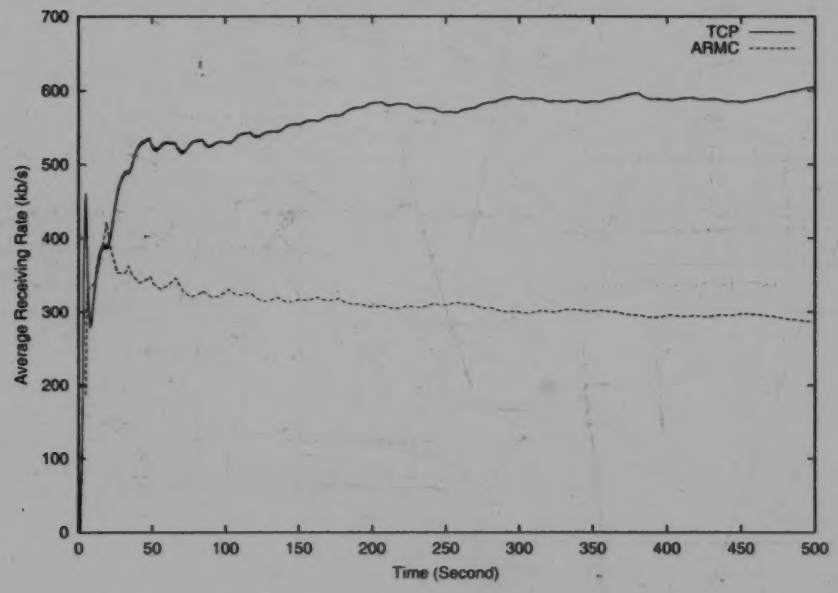


(c) $\alpha = 10$

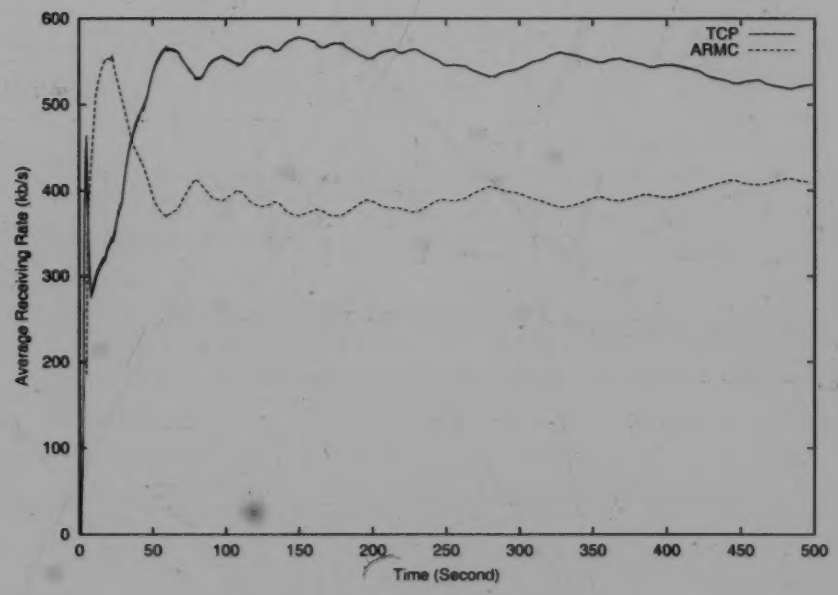


(d) $\alpha = 100$

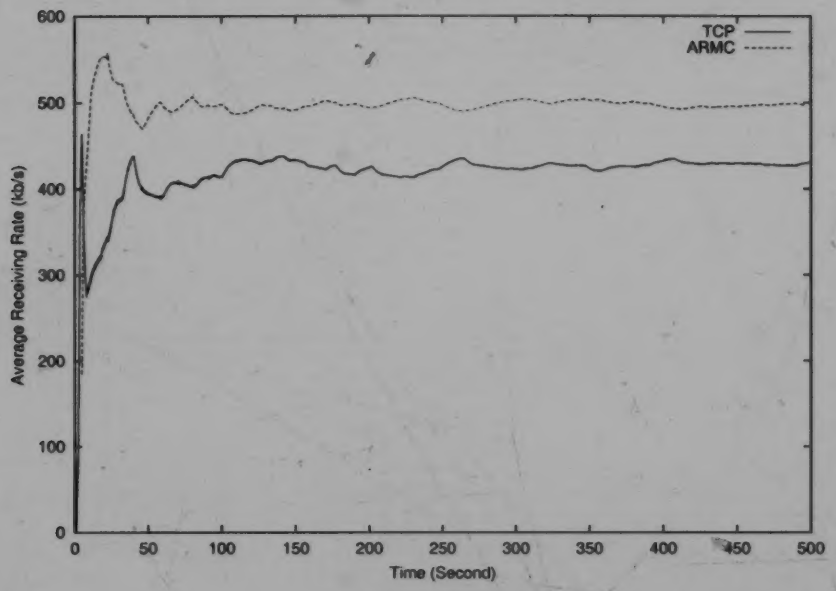
Figure 4.11: Receiving rate when a multicast flow and a TCP flow interact along the non-longest path. A multicast source has different RTT from the TCP flow.



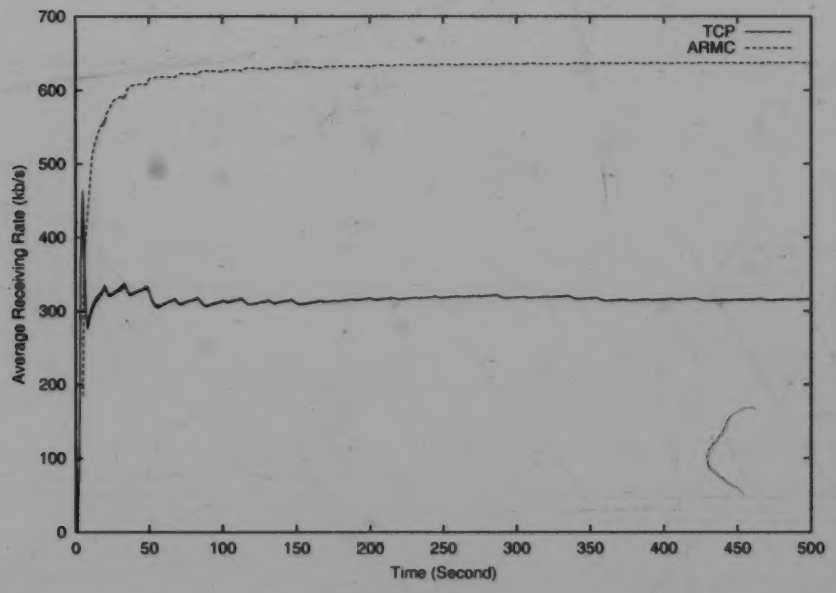
(a) $\alpha = 0$



(b) $\alpha = 2$



(c) $\alpha = 10$



(d) $\alpha = 100$

Figure 4.12: Average receiving rate when a multicast flow and a TCP flow interact along the non-longest path. A multicast source has different RTT from the TCP flow.

RTT than the multicast flow. These graphs also show that the multicast flow can grab more bandwidth than the TCP flow as the active router increases its buffers to smooth the congestion. The reason is similar to the first experiment. At that time the active router controls the data forwarding and rate regulation and its shorter RTT makes it grab more bandwidth.

In addition to the above experiments, we also conduct other experiments by varying the RTT of multicast flows and TCP flows. They all produce similar results. The experiments on a larger topology gives similar results as well.

4.4.2 Interaction among multiple multicast flows

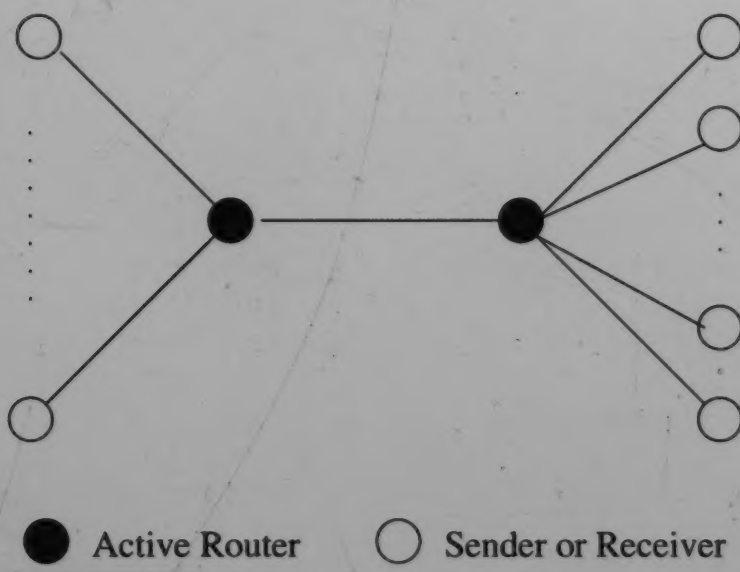


Figure 4.13: Topology

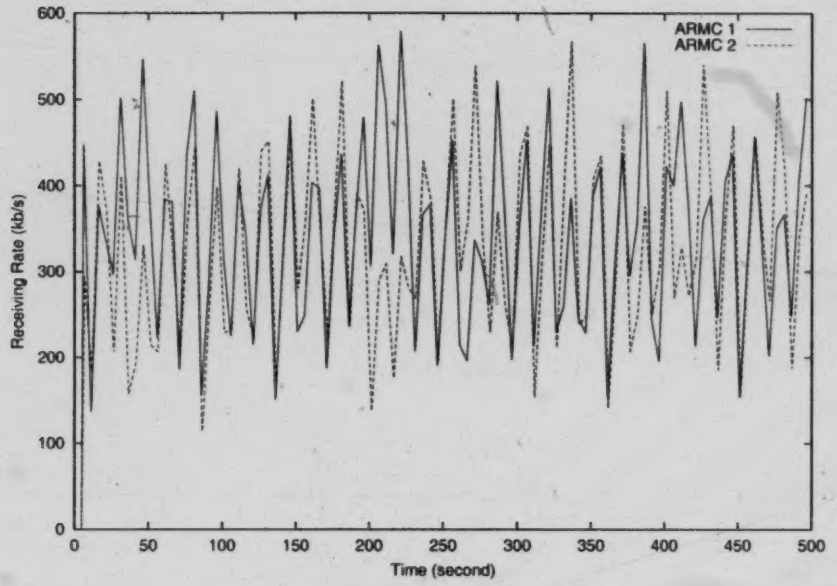
It is also important to explore the interaction among multiple multicast flows. Fig. 4.13 is the topology for evaluating the bandwidth competition among multiple multicast flows. Two multicast flows have the same RTT whose value is around 200ms. Each link has 1Mb bandwidth and uses a Droptail queue whose size is 50. Two kinds of experiments are conducted.

One is to investigate whether two multicast flows can compete for bandwidth fairly when they use the same amount of resources. The other is to investigate how two multicast flows can share the bandwidth when they use different amounts of resources. In both cases, two multicast flows only share the link between active routers and have the same number of receivers.

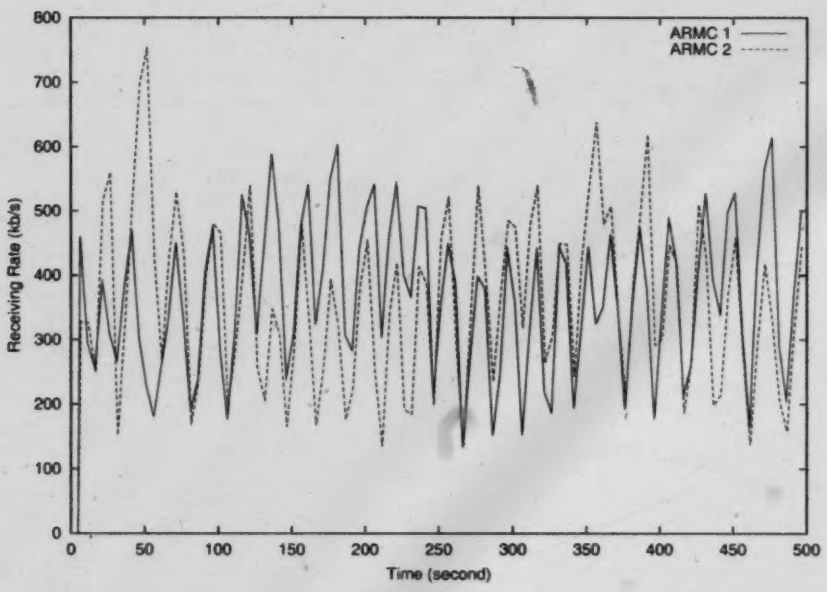
Fig. 4.14 and Fig. 4.15 are respectively Receiving Rate vs. Time and Average Receiving Rate vs. Time when two multicast flows uses the same amount of resources. That is, they have the same α value. They show that two multicast flows can fairly share the bandwidth of the congested link. This is expected since they use the same window regulation algorithm and use the same amount of resources.

Fig. 4.16 and Fig. 4.17 are respectively Receiving Rate vs. Time and Average Receiving Rate vs. Time when two multicast flows use different amounts of resources. They show that the greater the ratio between α_2 and α_1, the bigger the bandwidth that flow grabs. This is understandable since active routers for the flow with a larger buffer potentially have stronger capability to smooth the congestion and grab more bandwidth, and active routers for the flow with a smaller buffer have a larger probability of informing the source to slow down. Because the source usually has a longer RTT than active routers, the source can not increase the window as the active routers do. Note that the multicast flow is limited in grabbing a larger share of bandwidth by just increasing the buffer size. When the buffer size is increased to some value, an active router can accommodate all the congestion occurring on the downstream links without informing the upstream active routers or the source. Then it does not make a difference if the buffer size continues to increase since at that time the main factor to affect the bandwidth competition is RTT.

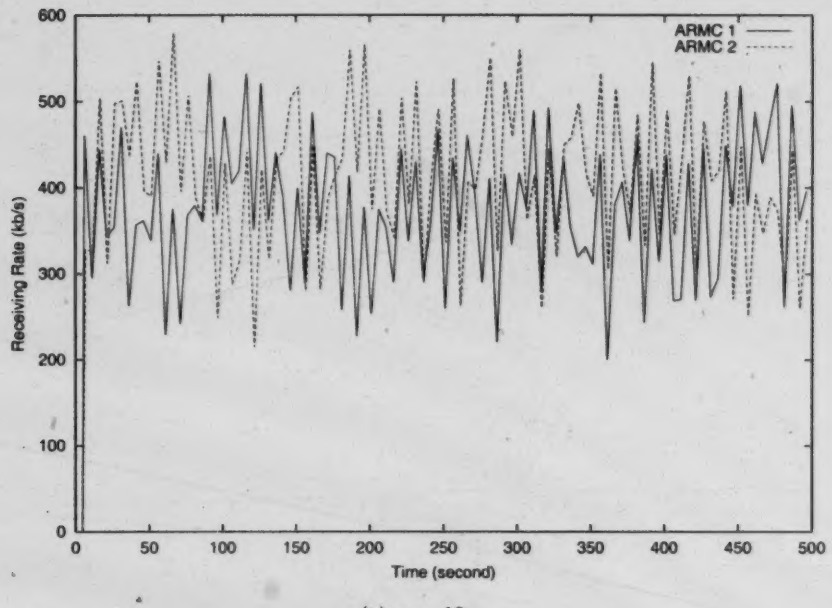
We also conducted the experiments by changing RTT of two multicast flows. They gave similar results.



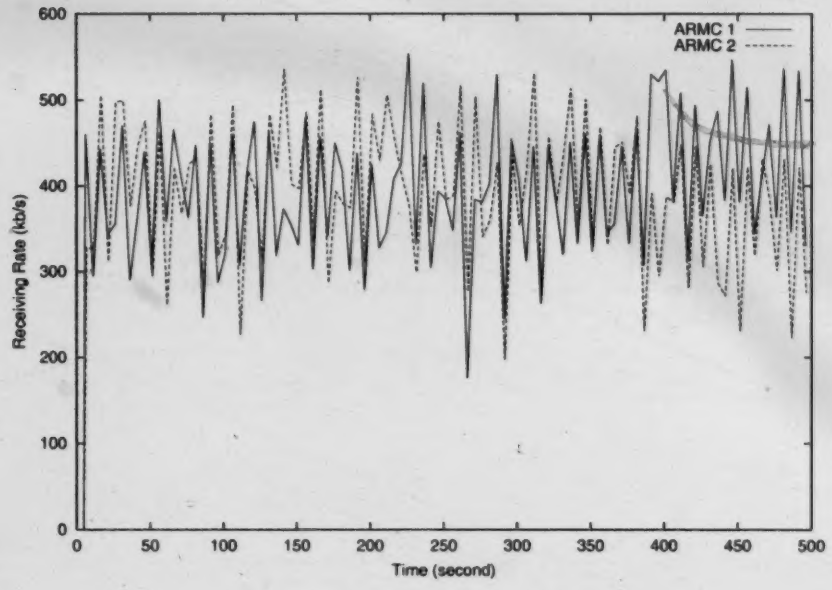
(a) $\alpha = 0$



(b) $\alpha = 2$

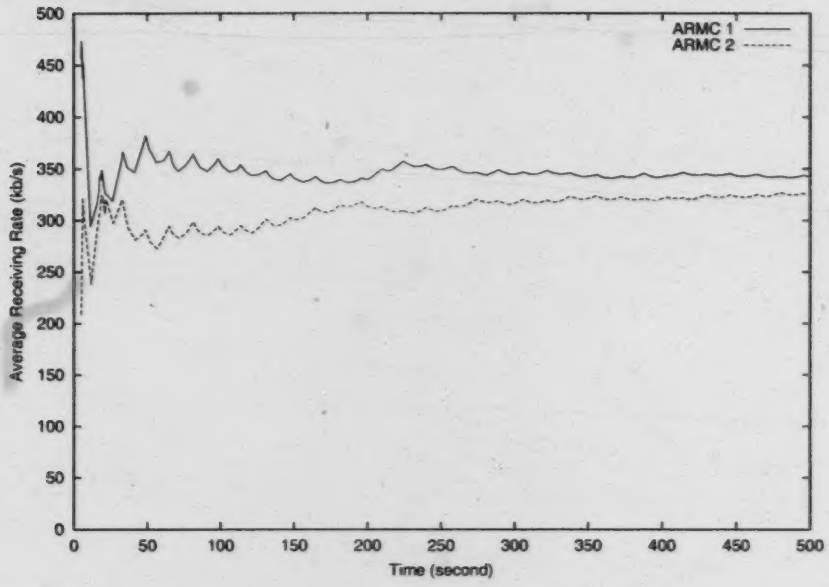


(c) $\alpha = 10$

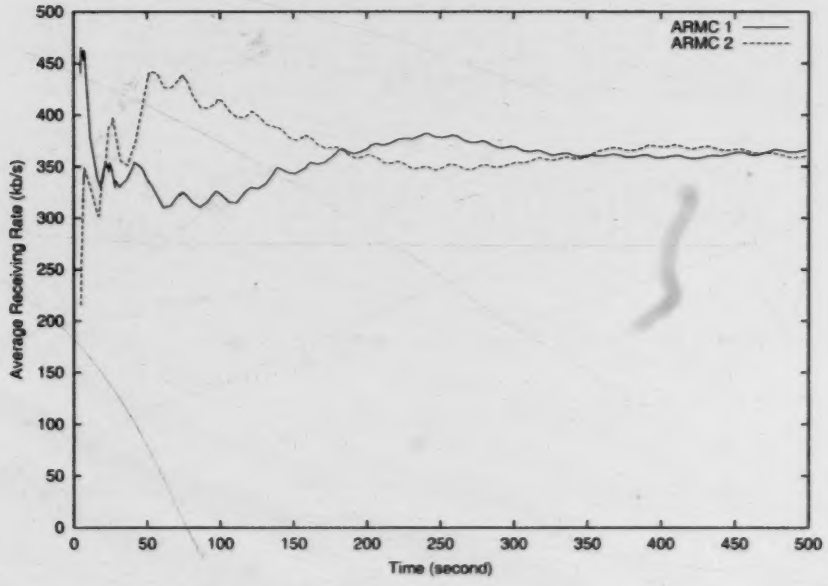


(d) $\alpha = 100$

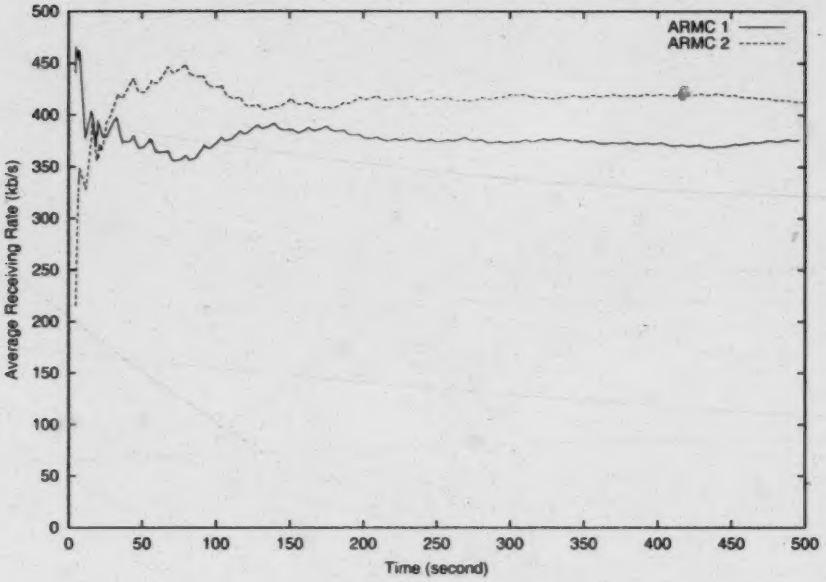
Figure 4.14: Receiving rate when two multicast flows use the same amount of resources.



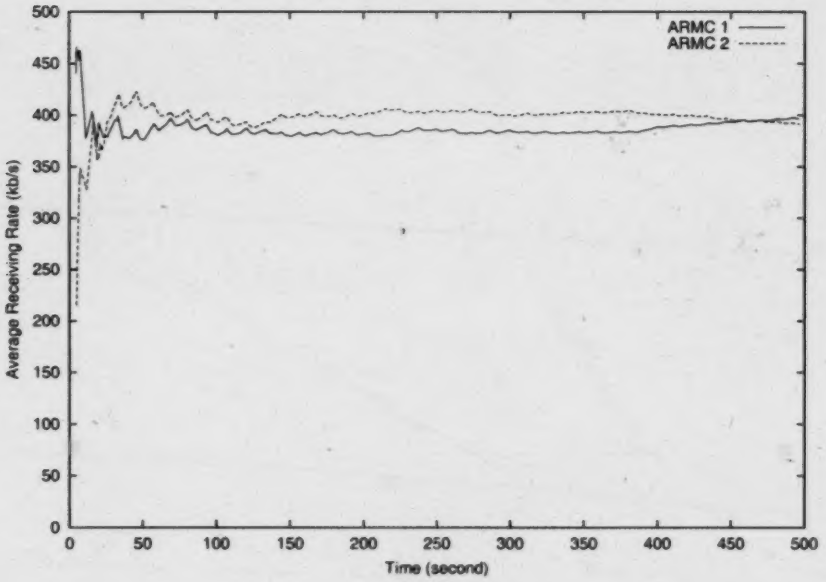
(a) $\alpha = 0$



(b) $\alpha = 2$

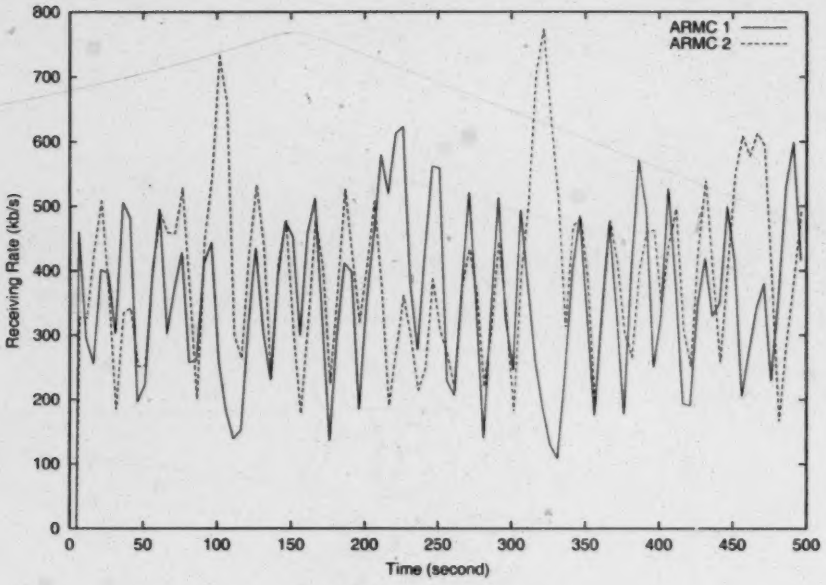


(c) $\alpha = 10$

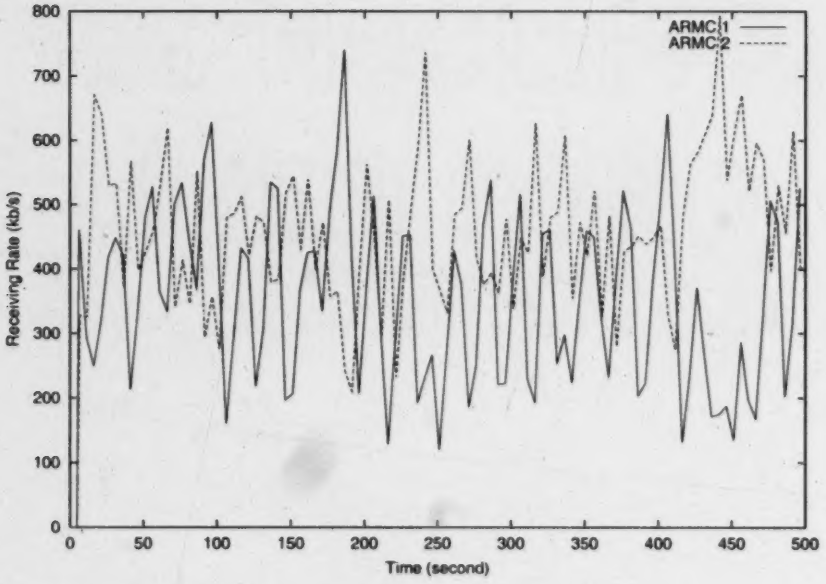


(d) $\alpha = 100$

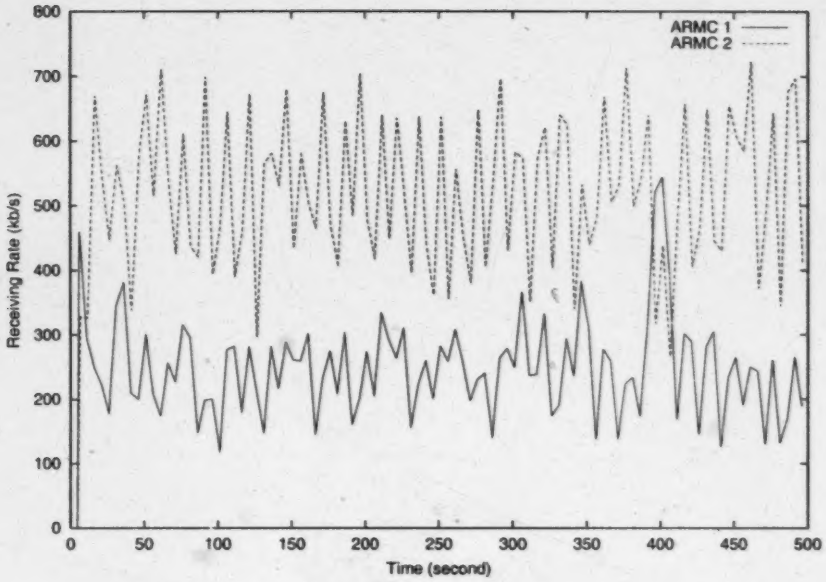
Figure 4.15: Average receiving rate when two multicast flows use the same amount of resources.



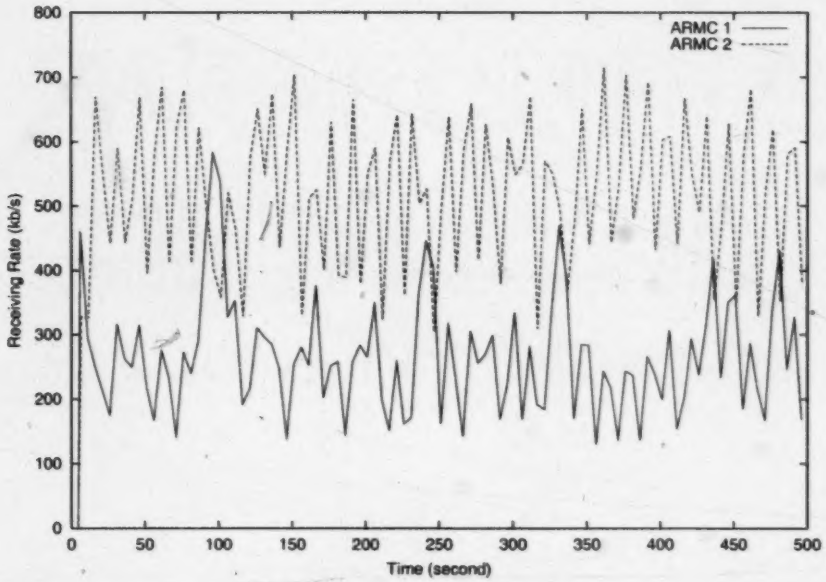
(a) $\frac{\alpha_2}{\alpha_1} = 2$



(b) $\frac{\alpha_2}{\alpha_1} = 4$

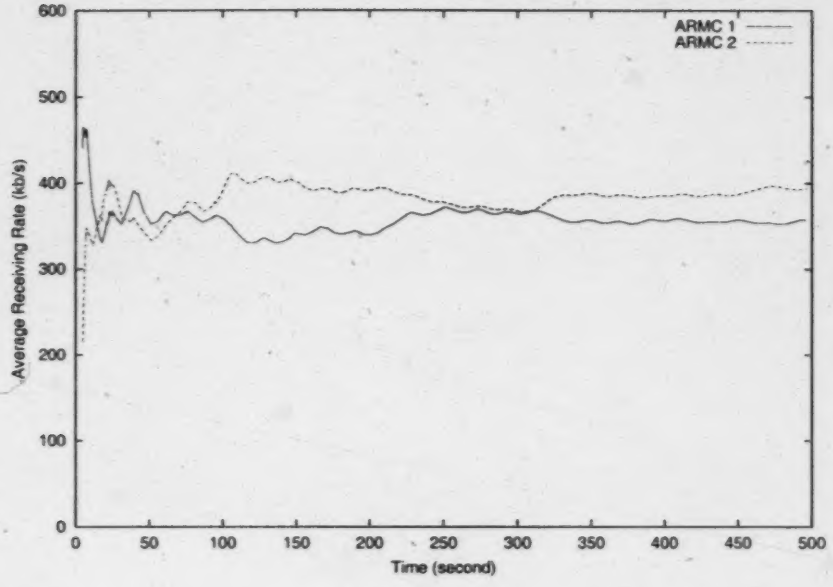


(c) $\frac{\alpha_2}{\alpha_1} = 10$

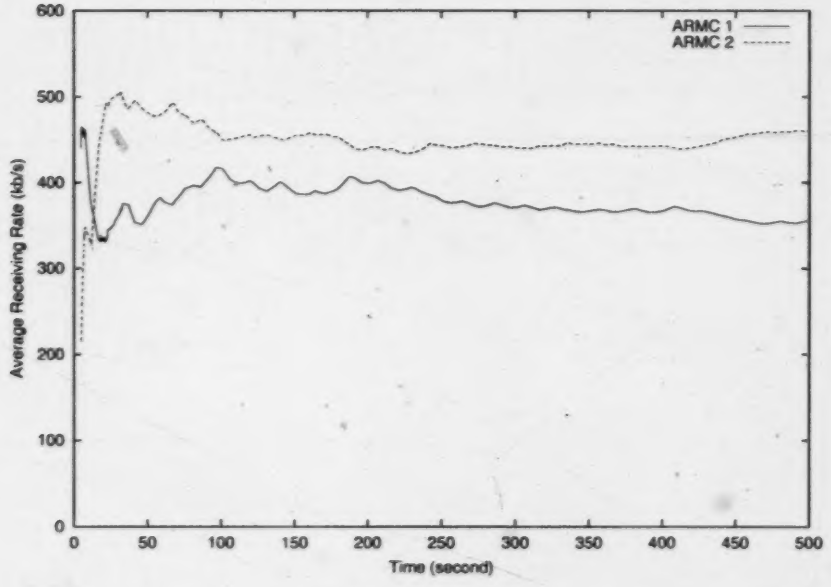


(d) $\frac{\alpha_2}{\alpha_1} = 100$

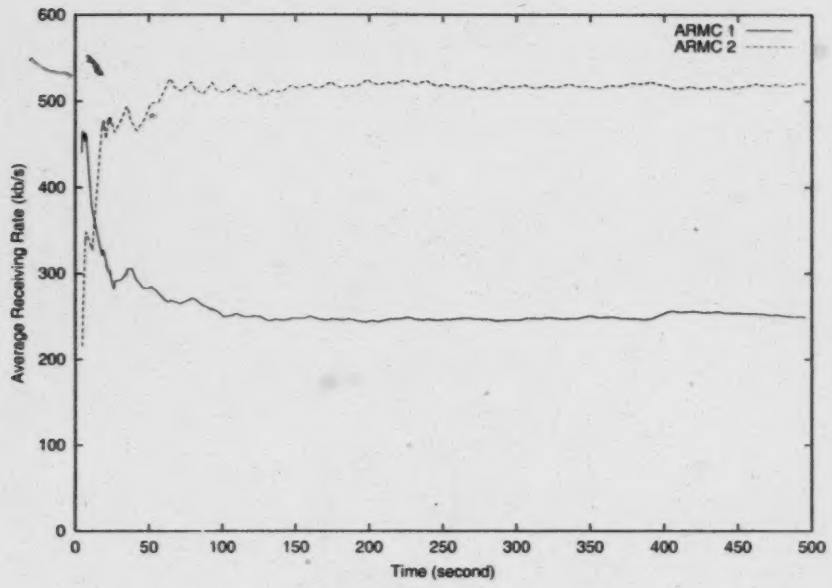
Figure 4.16: Receiving rate when two multicast flows use different amount of resources.



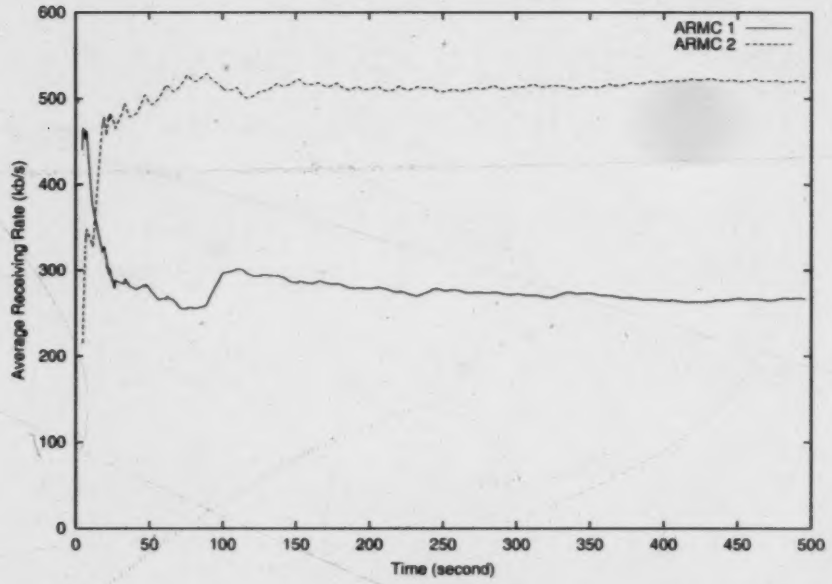
(a) $\frac{\alpha_2}{\alpha_1} = 2$



(b) $\frac{\alpha_2}{\alpha_1} = 4$



(c) $\frac{\alpha_2}{\alpha_1} = 10$



(d) $\frac{\alpha_2}{\alpha_1} = 100$

Figure 4.17: Average receiving rate when two multicast flows use different amount of resources.

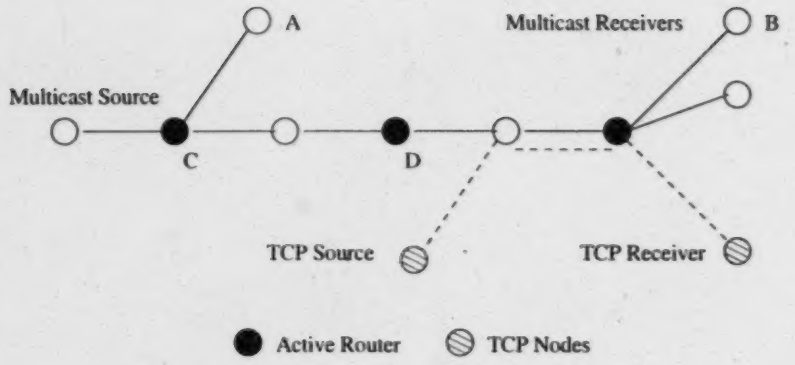
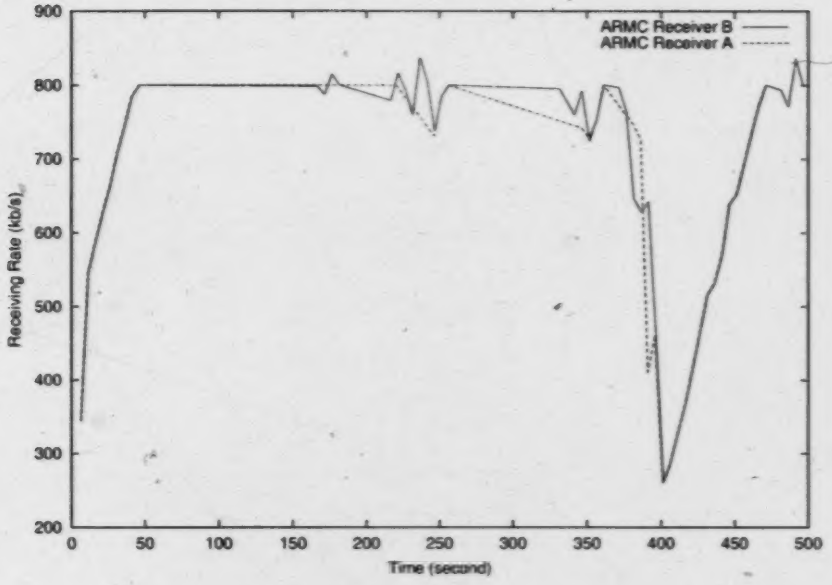


Figure 4.18: Topology

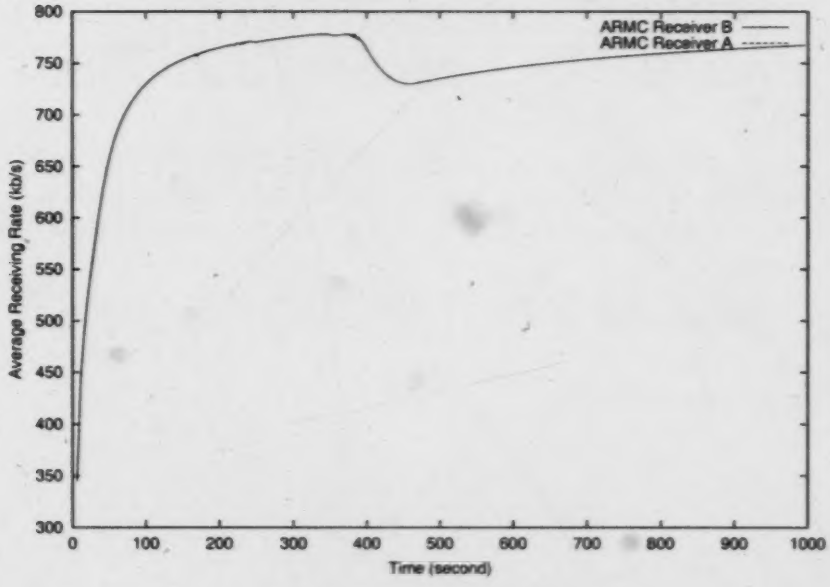
4.4.3 Inter-receiver fairness

Fig. 4.18 is the topology used for studying the inter-receiver fairness. Every link has 1 Mb bandwidth and uses a Droptail queue whose size is 30. In all the experiments, a multicast flow is modeled to have infinite data to send. The simulation lasts 1000 seconds. 5 TCP flows, each of which sends a number of packets randomly chosen between 10 kilobytes and 1000 kilobytes, randomly start and terminate between time 200 and time 600 to create temporary congestion. We investigate how active routers can use congestion smoothing buffers to accommodate the congestion and achieve inter-receiver fairness.

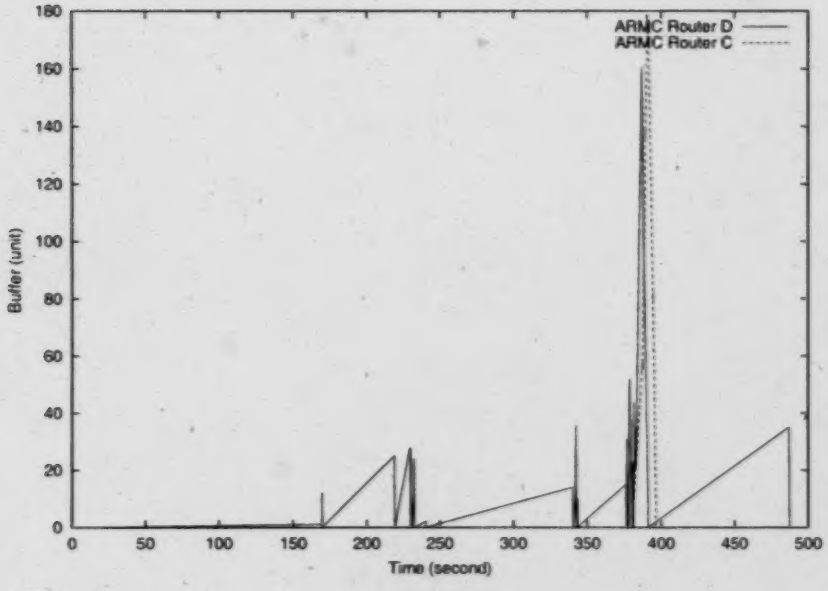
First we present the case where active routers can not smooth the congestion using its limited congestion smoothing buffer. Fig. 4.19(a) and Fig. 4.19(b) are respectively Receiving Rate vs. Time and Average Receiving Rate vs. Time for receiver A and B. Fig. 4.19(c) is Buffer vs. Time for active router C and D when α is 2. Since there are no significant changes after time 500, Fig. 4.19(a) and Fig. 4.19(c) only shows the receiving rate and buffer size evolution from time 0 to time 500. In Fig. 4.19(a), it seems abnormal that the receiving rate for receiver A is smaller than receiver B when congestion occurs around time 220 and active router D accommodates the congestion. However, it is reasonable since the congestion causes the packets to be built up in



(a) Receiving rate



(b) Average Receiving rate

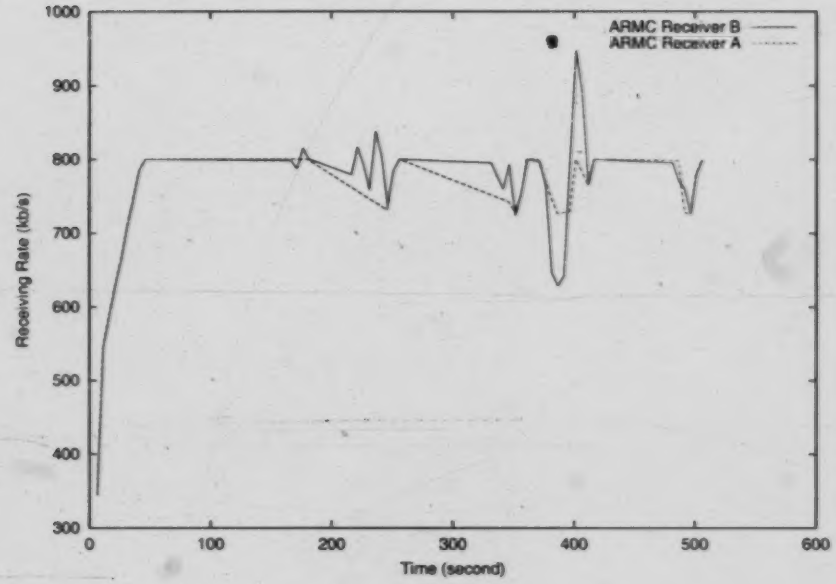


(c) Buffer

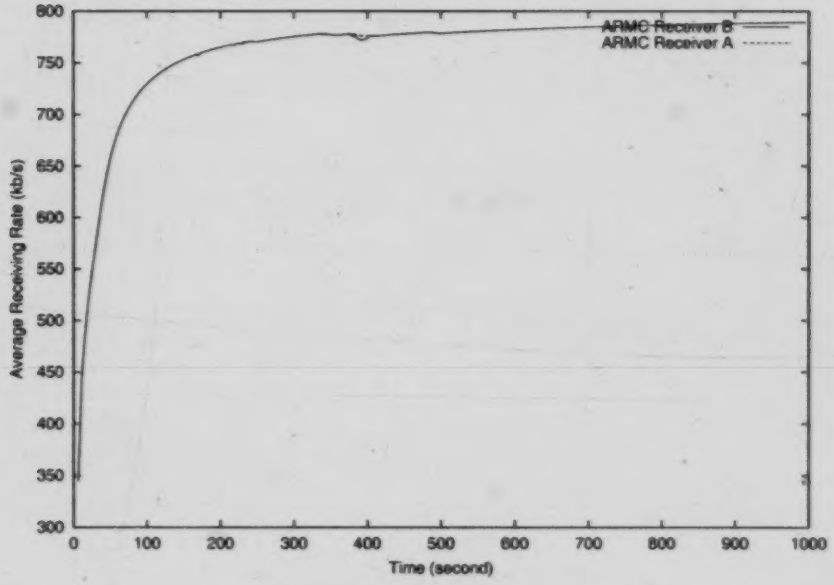
Figure 4.19: Inter-receiver fairness. $\alpha = 2$

the queue and finally the queuing delay is increased. Eventually the RTT perceived by the source becomes large. At the same time, the source has increased its congestion window to the maximal one. Consequently, when the RTT increases, the source slows down its sending rate. Although the congestion also can affect the RTT of active routers, it is not as significant as the source in this example. When the congestion is really heavy so that the active routers have no ability to accommodate it, the source is informed and reduces its rate to accommodate the congestion. The change of receiving rate in Fig. 4.19(a) around time 400 is caused by the heavy congestion. From Fig. 4.19(c), we can clearly see that the buffer size increases significantly around time 400 in active routers D and C. Fig. 4.19(b) shows that there is no big difference in the average receiving rate among different receivers for a long-term period since the congestion is bursty and all the static link capacity is the same.

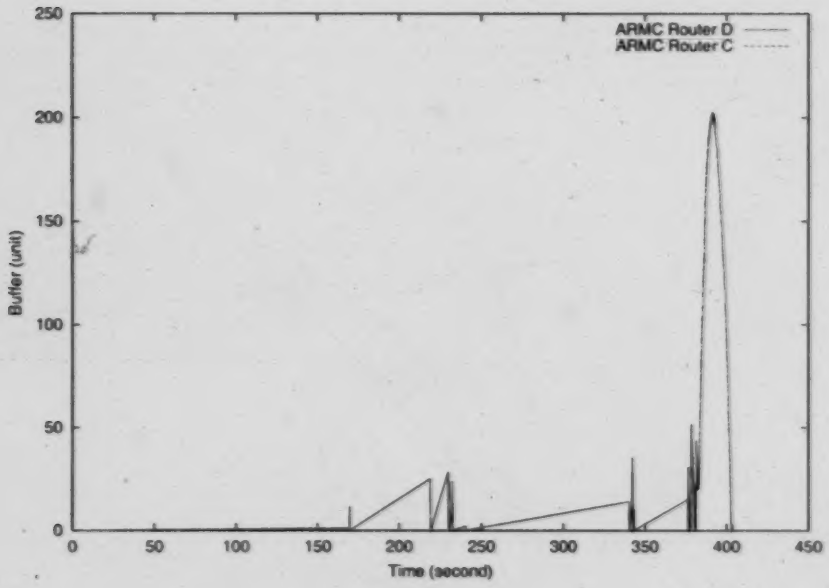
Next we present the case that active routers have enough capability to accommodate the congestion. Fig. 4.20(a) and Fig. 4.20(b) are respectively



(a) Receiving rate



(b) Average Receiving rate



(c) Buffer

Figure 4.20: Inter-receiver fairness. $\alpha = 4$

Receiving Rate vs. Time and Average Receiving Rate vs. Time for receiver A and B. Fig. 4.20(c) is Buffer vs. Time for active router C and D when α is 4. In Fig. 4.20(a), the rate difference between the receiver A and B around time 220 is caused by the same reason as the case where α is 2. When the heavy congestion happens around 370, the active router D has enough buffers to smooth it. Then, as it is shown in Fig. 4.20(a), the congestion only causes the rate reduction for receiver B. Receiver A only reduces rate slightly due to the increased RTT. After the congestion is gone, the active router D increases its rate and removes the buffered data. The buffer size change in Fig. 4.20(c) illustrates the change. When the congestion phase is terminated, the receiver A and B then have the same rate. Given enough buffer space in active routers, active routers can smooth the congestion and lessen the effect of congestion on unrelated receivers and then achieve the inter-receiver fairness to some degree.

The experiments on a large topology with 200 nodes generated by GT-ITM[KM97] give similar results.

4.4.4 Summary for congestion control

In this section, the performance evaluation for the congestion control scheme is presented in three parts: interaction with TCP flows, interaction among multiple multicast flows, and inter-receiver fairness. The simulation results show that the congestion control scheme can make a multicast flow obtain a different share of bandwidth as resources it uses change. In addition, the effect of congestion can be limited in a local region and inter-receiver fairness can be achieved to some degree.

4.5 Summary

In this chapter, we discuss the simulation results for the error recovery scheme from three perspectives: effect of retransmission buffer size on performance, effect of number of active routers on performance, and performance comparison among AMRC, ARM, and AER/NCA. They show that considerable performance can be achieved with limited support from routers and ARMC has a better performance than both ARM and AER/NCA in almost all the tested cases. Also, the results for the congestion control scheme are also presented in three perspectives: interaction with TCP flows, interaction among multiple multicast flows, and inter-receiver fairness. They show that ARMC can adjust its bandwidth share of the congested link as the resources which it uses change and also achieve the inter-receiver fairness to some extent.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have presented a novel active reliable multicast protocol. Its main contribution is incorporating an error recovery scheme and a congestion control scheme into a complete protocol. Our error recovery scheme innovates and integrates several simple but effective mechanisms into a practical reliable multicast protocol. It makes the error recovery more efficient by confining the scope of duplicate repair packets effectively. We use simulation to characterize the behavior of the protocol under the constraint of limited resources such as buffer space and the number of active routers. Comparison with two representative active reliable multicast protocols, ARM and AER/NCA, is also conducted. The results show that our error recovery scheme can achieve considerable performance improvement with only limited additional support from routers. The protocol outperforms ARM and AER/NCA in almost all the cases we tested.

More importantly, in a radical departure from contemporary popular end-to-end congestion control, we propose a router-assisted congestion control scheme for a reliable multicast protocol. Our congestion control scheme allows a multicast flow to get a different share of bandwidth by adjusting the resources bought from a network provider when competing with TCP flows. In addition, it can lessen the effect of congestion that occurs on some specific links on unrelated receivers and eventually achieve inter-receiver fairness to

some degree. Scalability and responsiveness can also be achieved. The preliminary results of performance evaluation show that those objectives are all attained with the help of active routers.

Our work further confirms that the active network concept and the end-to-end argument are not contradictory but complement each other. With proper design, router-assisted solutions could be more cost-effective than pure end-to-end approaches.

5.2 Future Work

There are several promising directions for future research.

- Evaluate performance of the protocol in a more sophisticated scenario. For example, we would like to use a more complex topology and change the number of receivers to evaluate our protocol. It is also an option to use the background traffic such as TCP or UDP flows to create loss for performance evaluation of the error recovery scheme.

- Implement ARMC in one of the active network platforms such as ANTS or SwitchWare[AH99] and investigate the processing cost at active routers.

- For the congestion control scheme, we plan to innovate our rate regulation algorithm so that our protocol behaves like an end-to-end congestion control protocol that can achieve TCP-friendliness for congestion occurring on any path when there is no buffer space in any active router. Furthermore, we will investigate the possibility of letting each active router maintain different forwarding rate for different downstream links rather than using one rate for the whole subtree so as to maximize the inter-receiver fairness while maintaining a reasonable cost.

- It is of interest to investigate the relationship between queue scheduling algorithms in routers and rate regulation algorithms for the congestion control algorithm.

- It would be useful to explore the opportunity for unreliable real-time applications such as streaming video to benefit from our protocol. Although unreliable real-time applications can tolerate the packet loss, the quality will be significantly degraded if the loss ratio is large. With the help of our error recovery scheme, it is possible that a large number of lost packets can be quickly retransmitted before they are forwarded to the applications and finally improve the quality perceived by applications. Our congestion control scheme can also be used to help the unreliable applications.

Bibliography

[AH99] A.T. Campbell, H.G. De Meer et al, "A Survey of Programmable Networks", *ACM SIGCOMM Computer Communication Review*, Vol. 29 No. 2 pg. 7-24, April 1999.

[AJ01] A. Legout, J. Nonnenmacher, E.W. Biersack, "Bandwidth Allocation Policies for Unicast and Multicast Flows", *IEEE/ACM Transactions on Networking*, Aug. 2001.

[AS99] A.M. Costello and S. McCanne, "Search Party: Using Randomcast for Reliable Multicast with Local Recovery" *Proc. of Infocom'99*, New York, March 1999.

[BJ98] B.N. Levine and J.J. Garcia-Luna-Aceves, "A Comparison of Reliable Multicast Protocols," *ACM Multimedia Systems Journal*, Vol. 6, No.5, August 1998. pp334-348.

[BJ01] B. Bai, J. Harms, and Y. Li, "Active Error Recovery for Reliable Multicast", *Proc. of ICCCN'01*, Oct. 2001

[CG98] C. Papadopoulos, G. Parulkar, and G. Varghese, "An Error Control Scheme for Large-scale Multicast Applications", *Proc. of IEEE INFOCOM'98*, San Francisco, CA, March 1998.

[DJ99] D. Rubenstein, J. Kurose, and D. Towsley, "The Impact of Multicast Layering on Network Fairness", *Proc. of ACM SIGCOMM'99*, Cambridge, MA, September 1999.

[DK98] D. Delucia and K. Obraczka, "A Multicast Congestion Control Mechanism for Reliable Multicast", *Proc. of ISCC*, June, 1998.

[DR87] D. Bertsekas and R. Gallager, "Data Networks", *Prentice Hall*, 1987.

[FA98] F. P. Kelly, A.K. Maulloo, and D.K.H. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability", *Journal of the Operational Research Society*, vol. 49(3):237-252, March 1998.

[FK97] F. Kelly, "Charging and rate control for elastic traffic", *European Transactions on Telecommunications*, Vol. 8, pp. 33-37, 1997.

[HD84] H. Saltzer, D. P. Reed, and D. Clark, "End-to-end Arguments in System Design", *ACM Transactions on Computing Systems*, vol. 2, no. 4, 1984.

[HM98] H.A. Wang, and M. Schwartz, "Achieving Bounded Fairness for Multicast and TCP Traffic in the Internet", *Proc. of ACM SIGCOMM'98*, September 1998.

[HS95] H.W. Holbrook, S. K. Singhal, and D. R. Cheriton, "Log-based Receiver-reliable Multicast for Distributed Interactive Simulation", *Proc. of ACM SIGCOMM'95*, pp. 328-341, October 1995.

[JJ00] J. Crowcroft, J. Gemmell et al., "PGM Reliable Transport Protocol Specification", *Internet draft*, work in progress, April, 2000.

[JP95] J.S. Ahn, P.B. DanZig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: Emulation and Experiments", *Proc. of SIGCOMM'95*, Oct. 1995

[JR99] J. Mo, R.J. La, V. Anantharam, and J. Walrand, "Analysis and Comparison of Reno and Vegas", *Proc. of IEEE Infocom'99*, New York, USA, March 1999.

[JS97] J. Lin, S. Paul, K. Sabnani, and S. Bhattacharyya, "A Reliable Multicast Transport Protocol(RMTP)", *IEEE Journal on Selected Areas in Communications*, Vol.15, April 1997.

[KM97] K. Calvert, M. Doar and E.W. Zegura, "Modeling Internet Topology", *IEEE Communications Magazine*, June 1997.

[KS99] K.-W. Lee, S. Ha and V. Bharghavan, "IRMA: A Reliable Multicast Architecture for the Internet", *Proc. of IEEE INFOCOM'99*, 1999.

[LL98] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like Congestion Control for Layered Multicast Data Transfer", *Proc. of IEEE INFOCOM*, San Francisco, CA, April 1998.

[LN99] L. Rhee, N. Balaguru, and G.N. Rouskas, "MTCP: Scalable TCP-like Congestion Control for Reliable Multicast", *Proc. of IEEE INFOCOM'99*, New York, USA, March 1999.

[LR00] L. Rizzo, "pgmcc: a TCP-friendly Single-rate Multicast Congestion Control Scheme", *Proc. of SIGCOMM*, 2000.

[LS98] L. Lehman, S. Garland and D. Tennenhouse, "Active Reliable Multicast", *Proc. of IEEE INFOCOM'98*, 1998.

[MH97] M. Handley, "An Examination of MBone Performance", *USC/ISI Research Report: ISI/RR-97-450.*

[MJ96] M. Yajnik, J. Kurose, and D. Towsley, "Packet Loss Correlation in the MBone Multicast Network", *Proc. of IEEE Global Internet'96*, London, U.K., November 1996.

[MJS96] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP Selective Acknowledgment Options", *RFC 2018*, April, 1996.

[MJ00] M. Vojnovic, J. Y. Le Boudec, and C. Boutremans "Global fairness of additive-increase and multiplicative-decrease with heterogeneous round-trip times", *Proceedings of IEEE INFOCOM'2000*, Tel Aviv, Israel, March 2000.

[MS99] M. Handley and S. Floyd, "Strawman Specification for TCP Friendly (Reliable) Multicast Congestion Control (TFMCC)", *RMRG Meeting*, 1999.

[MSJ99] M. Yajnik, S.B. Moon, J. Kurose, and D. Towsley, "Measurement and Modeling of the Temporal Dependence in Packet Loss", *Proc. of IEEE INFOCOM'99*, New York, USA, March 1999.

[NS01] N. F. Maxemchuk and S. H. Low, "Active Routing", *IEEE Journal on Selected Areas in Communications*, 2001

[NSV2] http://www.isi.edu/nsnam/ns/.

[PANA] PANAMA Project, http://www.tascnets.com/panama/

[RJ95] R. Yavatkar, J. Griffioen, and M. Sudan, "A Reliable Dissemination Protocol for Interactive Collaborative Applications", *ACM Multimedia*, 1995.

[SD90] S. Deering and R. Hinden, "Multicast Routing in Datagram Internetworks and Extended LANs", *ACM Transactions on Computer Systems*, 8(2), May 1990

[SD99] S. Bhattacharyya, D. Towsley, J. Kurose, "The Loss Path Multiplicity Problem for Multicast Congestion Control", *Proc. of IEEE INFOCOM'99*, New York, USA, March 1999.

[SJ98] S.Bhattacharyya, J. Kurose, D. Towsley, and R. Nagarajan, "Efficient Rate-Controlled Bulk Data Transfer using Multiple Multicast Groups", *Proc. of Infocom'98*, San Francisco, CA, April 1998.

[SK97] S. Bhattacharjee, K. Calvert and E. Zegura, "Active Networking and the End-to-End Argument", *Proc. of ICNP '97*, Atlanta, GA, October 1997.

[SK99] S. Floyd, and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, August 1999.

[SKG99] S.J. Golestani and K.K. Sabnani, "Fundamental Observations on Multicast Congestion Control in the Internet", *Proc. of IEEE INFOCOM'99*, New York, USA, March 1999.

[SKV01] S. Ha, K. Lee and V. Bharghavan, "A Simple Mechanism for Improving the Throughput of Reliable Multicast", *Proc. of IEEE INFOCOM'01*, April, 2001.

[SM00] S. Shi and M. Waldvogel, "A Rate-based End-to-end Multicast Congestion Control Protocol", *Proc. of ISCC 2000*, July 2000.

[SS00] S.K. Kasera, S. Bhattacharyya et al., "Scalable Fair reliable Multicast Using Active Services", *IEEE Network Magazine (Special Issue on Multicast)*, Jan/Feb, 2000.

[SV95] S. Floyd, V. Jacobson, S. McCanne, C.-G. Liu, and L. Zhang, "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing", *Proc. of ACM SIGCOMM'95*, pp. 342-356, October 1995.

[SV96] S. McCanne and V. Jacobson, "Receiver-driven Layered Multicast", *Proc. of ACM SIGCOMM'96*, Aug, 1996.

[TE99] T. Jiang, E.W. Zegura, and M. Ammar, "Inter-Receiver Fair Multicast Communication over the Internet", *Proc. of of NOSSDAV'99*, AT&T Research Labs, Basking Ridge, New Jersey, June 1999.

[TF98] T. Faber, "ACC: Active Congestion Control", *IEEE Network*, May/June 1998, pp. 61-65.

[TM98] T. Jiang, M.H. Ammar and E.W. Zegura, "Inter-Receiver Fairness: A Novel Performance Measure for Multicast ABR Sessions", *Proc. of ACM SIGMETRICS'98*, Madison, Wisconsin, June 1998.

[TU97] T.V. Lakshman and U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-delay Products and Random Loss", *IEEE/ACM Transactions on Networking*, 5(3):336-350, June 1997.

[VJ88] V. Jacobson, "Congestion Avoidance and Control", *Proc. of ACM SIGCOMM'98*, , Aug. 1988.

[XS98] X. Li, S. Paul, P. Pancha, and M. Ammar, "Layered Video Multicast with Retransmissions (LVRM): Evaluation of Hierarchical Rate Control", *Proc. of Infocom'98*, 1998.

[YC01] Yuan Gao and Chao-Ju Hou, "RACCOOM: a rate-based congestion control scheme for multicasts", *SPIE Conference on Scalability and Traffic Control*, August 2001.

[YY00] Y. Gao, Y. Ge, and J. C. Hou, "Reliable Multicast for Core-Based Multicast Routing", *IEEE 8th Int'l Conf. on Network Protocols (ICNP'2000)*, November 2000.

[ZK01] Z. Xiao and K.P. Birman, "A Randomized Error Recovery Algorithm for Reliable Multicast", *Proc. of IEEE INFOCOM'01*, 2001.